Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Modulární výukové prostředí pro IP telefonii na platformě OpenWRT

Modular Learning Environment for IP telephony Based on OpenWRT Platform

2018 **Bc. Ondřej Kocurek**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Ondřej Kocurek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2601T013 Telekomunikační technika

Téma: Modulární výukové prostředí pro IP telefonii na platfromě OpenWRT
Modular Learning Environment for IP telephony Based on OpenWRT Platform

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem diplomové práce je seznámit se a následně vytvořit sérii šablon pro automatickou konfiguraci IP telefonních prvků na platformě OpenWRT za účelem nasazení řešení do výuky.

Zadání:

1. Detailně nastudujte možnosti nasazení IP telefonních nástrojů na platformě OpenWRT.
2. Navrhněte koncept pro automatickou konfiguraci IP telefonního řešení na základě vložených šablon.
3. Navrhněte a vytvořte základní konfigurační šablonu a implementujte grafické rozhraní pro správu prostředí.
4. Otestujte vytvořené schéma v laboratorních podmínkách.
5. Vytvořte dokumentaci k výukovému prostředí a postup k vytváření šablon.

Seznam doporučené odborné literatury:

1. Christopher Hallinan, Embedded Linux Primer: A Practical Real-World Approach (2nd Edition), ISBN-13: 978-0137017836.

2. Russel Bryant, Leif Madsen, Jim Van Megellen, Asterisk: The Definitive Guide, ISBN-13: 978-1449332426

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Filip Řezáč, Ph.D.**

Datum zadání: 01.09.2016

Datum odevzdání: 30.04.2018

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky ID16 OSTRAVA

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *24. dubna 2018*

.....

podpis studenta

Poděkování

Rád bych velice poděkoval Ing. Filipu Řezáči, Ph.D. za velkou odbornou pomoc, konzultaci, užitečné rady a za projevenou trpělivost při vytváření této diplomové práce.

Abstrakt

Cílem této diplomové práce je navrhnout a vytvořit řešení, které by mělo praktické využití při výuce předmětů souvisejících s IP telefonií na VŠB-TUO. Mělo by se jednat o řešení, které v sobě spojuje jednoduchost, efektivnost a hlavně praktičnost. Cílem je vytvořit modulární systém, který by byl implementován na stávající infrastrukturu používanou při výuce předmětů souvisejících s IP telefonií, a ušetřit tak čas jak vyučujícímu, tak studentům. Nynější situace vyžaduje časově náročnou konfiguraci ze strany studentů a hlavně vysokou časovou náročnost pro vyučující při hledání chyb v jejich konfiguraci. Tudíž ne vždy zůstává dostatek času na ukázání praktického využití nakonfigurovaných systémů dle zadaných úkolů. Pro dosažení stanovených cílů je využit operační systém OpenWRT/LEDE, který využívá pro správu grafické rozhraní Luci. Žádoucí je vytvořit rozšíření pro toto grafické prostředí, které by vyučujícím v mnohém zjednodušilo práci a umožnilo automatickou konfiguraci vzdálených stanic studentů tak, aby byly připraveny pro zadané úkoly. K tomuto účelu je předpoklad využití jedné ze služeb, či balíčků uciprov, R-sync či scp.

Klíčová slova

OpenWRT; LEDE; LuCI; Lua; Asterisk; R-sync; Scp; Uciprov; IP telefonie; SIP; CBI

Abstract

The goal of this work is to design and create a solution, which should have a practical usage during teaching of IP telephony related subjects. It should be a solution that combines simplicity, efficiency and practicality. The goal is to create a modular system, which could be implemented on an existing infrastructure that is used to teach IP telephony related subjects; therefore it should save time for both the teachers and students. The current situation requires a time-consuming configuration from the perspective of the students and mainly a high amount of time for teachers to find errors in these students´ configurations. Therefore there is not always enough time to show the practical application of the configured systems according to the assigned tasks. To achieve the set goals, the OpenWRT/LEDE operation system is used which has graphical interface named Luci. It is desirable to create an extension for this graphic environment that would make it easier for teachers to work and allow configuring remote student stations automatically, so that they would be prepared for the assigned tasks. For this purpose, it is assumed to use one of these packages or services, uciprov, R-sync or scp.

Key words

OpenWRT; LEDE; LuCI; Lua; Asterisk; R-sync; Scp; Uciprov; IP telefonie; SIP; CBI

Seznam použitých zkratek

Zkratka	Význam
B2BUA	back-to-back user agent
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
HTML	HyperText Markup Language
IP	Internet Protocol
IVR	Interactive Voice Response
NAT	Network Address Translation
PBX	Private Branch Exchange
QoS	Quality of Service
RTCP	RTP Control Protocol
RTP	Real-time Transport Protocol
SIP	Session Initiation Protocol
SSH	Secure Shell
TCP	Transmission Control Protocol
UCI	Unified Configuration Interface
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
VoIP	Voice over Internet Protocol
VŠB-TUO	Vysoká škola báňská - Technická univerzita Ostrava

Obsah

Seznam obrázků

Seznam tabulek

Úvod

V rámci výuky odborných předmětů zaměřených na problematiku technologie VoIP, především během výuky konfigurace jednotlivých prvků v IP telefonní infrastruktuře, dochází velice často k výrazným časovým prodlevám vlivem chybné konfigurace, a to například díky překlepům, nepozornosti, nezkušenosti či nedodržování zadaného postupu ze strany studentů. Během výuky se tudíž spousta času věnuje odstraňování chyb v konfiguraci SIP prvků, vyučující je proto nucen tyto chyby řešit individuálně, a tím výuka poněkud zadrhává. Představou je, aby se jednotlivé předpřipravené scénáře různých situací a přednastavených služeb vzdáleně přenesly na konfigurovaný prvek, tudíž by se omezila možnost chyby na straně uživatele a studenti by se mohli zabývat již konkrétní problematikou a zadaným úkolem: například sledovat SIP komunikaci, díky čemuž by bylo mnohem více času na tyto praktické ukázky a pokročilou konfiguraci. Pro vytvoření uživatelsky přívětivé a modulární platformy pro hromadnou správu konfigurace byl zvolen firmware OpenWRT/LEDE, na kterém by běžel SIP B2BUA Asterisk, kdy pro výběr tohoto prvku hovoří výhody uvedené dále v textu. Hlavním problémem, kterému je potřeba čelit je, jakým způsobem by bylo možné provádět vzdálenou správu konfigurace SIP B2UA Asterisk, a to rychle a jednoduše. Jelikož Asterisk má několik konfiguračních souborů různě umístěných, které je potřeba pro různé scénáře upravovat, je žádoucí nalézt řešení, které umožní jednoduché a bezpečné přepsání těchto konfiguračních souborů dle přání uživatele. V rámci návrhu konceptu bylo uvažováno nad několika možnostmi, jak vše vyřešit. Jako první možnost a zároveň nejelegantnější řešení se jevilo použití rozhraní pro hromadnou správu - Uciprov, které umožňuje zautomatizovanou vzdálenou správu různých embedded zařízení (primárně směrovačů). Druhou možností se jevilo použití R-sync a jako poslední možnost příkaz scp. Tato navrhovaná řešení však pokrývala pouze konfigurační část, bylo potřeba také vyřešit uživatelsky přívětivou možnost, jak vše spravovat a ovládat. Volba padla na webové rozhraní, které bylo vytvořeno pro OpenWRT, a tím je LuCI. V první části této diplomové části naleznete základní teoretické informace o jednotlivých výše zmíněných termínech. V druhé části se již věnuji implementaci a praktickému použití, stejně tak i problémům, na které jsem narazil a které jsem úspěšně či neúspěšně vyřešil. Tato část by měla sloužit také jako návod pro někoho dalšího, kdo by po mě chtěl vytvořit další scénáře, a tím rozšířit možnosti použití tohoto řešení. V přílohách je možné nalézt uživatelskou dokumentaci, která by měla sloužit jako návod popisující, jak s vytvořeným řešením pracovat. Dále také programovou dokumentaci.

1 OpenWRT/LEDE

1.1 Historie

Projekt OpenWRT vznikl v lednu 2004 vydáním jeho první verze určené pro směrovače Linksys WRT54G, jenže firmware, který firma Linksys použila, byl založen pod licencí GNU (více o GNU licenci [1]). Tudíž dle pravidel GNU licencí museli svůj zdrojový kód zveřejnit. Toho využili jiní vývojáři a firmware OpenWRT se začal rozšiřovat i na jiná zařízení. Během zpracování této diplomové práce byla po většinu času používána v té době nejaktuálnější dostupná verze 15.05.1 Chaos Calmer, kdy pojmenování verzí vycházelo z názvů alkoholických koktejlů. Avšak s rokem 2018 se projekt OpenWRT spojil s bývalým projektem LEDE a společně vydali novou verzi s názvem LEDE 17.01.4.

Obrázek 1.1: Ukázka OpenWRT Chaos Calmer

1.2 Obecné informace

OpenWRT/LEDE je vysoce rozšiřitelná GNU/Linux distribuce, která je primárně určená pro různá embedded zařízení (nejčastěji směrovače), přičemž počet zařízení, na kterých je možné tento firmware použít, se stále rozrůstá. Mezi jeho hlavní výhody patří vysoká modulárnost. Díky této vlastnosti umožňuje vysokou flexibilitu použití a nasazení i na méně výkonných zařízeních. Pro instalování balíčků využívá OpenWRT/LEDE pong balíčkový systém, díky kterému je správa softwarových balíčků rychlá a jednoduchá. Aktuálně je skrze repositáře dostupných téměř 2000 binárních balíčků.

Obrázek 1.2: Logo OpenWRT

Pro správu OpenWRT/LEDE se používá příkazový řádek BusyBox ash (více informací [2]) nebo webové uživatelské rozhraní. Druhá možnost je od verze 8.09 mnohem přívětivější pro uživatele a je založena na Framework LuCI.

1.3 LuCI

1.3.1 Historie

Příběh projektu LuCI se začal psát v březnu 2008 jako projekt Fleci coby součást snahy o přenesení Freifunk-Firmware z verze OpenWRT White Russian (v0.9) na verzi Kamikaze (v7.06-8.09.2). Prvotním důvodem pro vytvoření tohoto projektu byla absence jednoduchého, bezplatného, lehce rozšiřitelného a také snadno udržovatelného webového prostředí pro správu různých embedded zařízení. Autoři projektu se rozhodli použít programovací jazyk Lua. Ten používá objektově orientované programování. Více informací je uvedeno v 1.3.2. LuCI má rozdělenou logickou strukturu na jednotlivé části, jako jsou models, views a jiné. Díky tomu se zjednodušila údržba, snížil potřebný výkon, zmenšil požadovaný prostor pro instalaci, díky čemuž se dosáhlo i rychlejšího běhu celého rozhraní. Toto řešení rozhodně dosáhlo úspěchu, jelikož se stalo součásti distribuce OpenWRT od zmíněné verze Kamikaze.

1.3.2 Lua

Programovací jazyk Lua byl navržen, vyvinut a je i nadále spravován týmem z Pontifical Catholic University of Rio de Janeiro v Brazílii. Jméno Lua (správná výslovnost je „LOO-ah“) znamená v portugalštině měsíc, konkrétně je to míněno jako měsíc, který krouží kolem Země.

Lua je výkonný, nenáročný a nezávislý skriptovací jazyk, který podporuje procedurální programování, objektově orientované programování aj. Lua kombinuje jednoduchou procedurální syntaxi s výkonnými konstrukcemi popisu dat založenými na asociativních polích a rozšiřitelnou sémantiku. Jednou z hlavních výhod tohoto programovacího jazyku je jeho rychlost, dle autorů se jedná o jeden z nejrychlejších skriptovacích jazyků. Další výhodou je univerzálnost, jelikož distribuce probíhá v malém balíčku a je možné ji použít na všech platformách, ať už se jedná o Unix, Windows nebo mobilní zařízení, tak třeba i integrovaných mikroprocesorech (příklad ARM nebo Rabbit), či sálových počítačích. Lua se také pyšní velice dobře zdokumentovaným API a rychlým jazykovým engine, díky čemuž je jednoduchá jeho integrace do kódu psaného i v jiných jazycích. Jistě další nespornou výhodou je, že Lua je šířena pod open source licencí (licence MIT [3]), což obecně znamená možnost používání k jakémukoliv účelu včetně komerčního. To, že se programovací jazyk osvědčil v praxi, dokládá i jeho používání v mnoha průmyslových aplikacích (např. Adobe Photoshop Lightroom) s důrazem třeba i na hry (např. World of Warcraft či Angry birds).

Nejnovější verzí je verze 5.3.4. vydaná 30.1.2017.

Obrázek 1.3: Logo Lua

1.4 Nástroje pro synchronizaci obsahu a vzdálenou správu

1.4.1 Uciprov

Jedná se o vedlejší efekt vývoje projektu BEESIP. Jde o nástroj, který umožňuje vzdálenou správu jakéhokoliv zařízení, na kterém běží OpenWRT. Tento projekt je však stále ve vývoji. Primárně byl vytvořen pro automatickou konfiguraci zařízení v rámci sítě Eduroam. Jádro systému umí automaticky z obecné adresy URI stáhnout a nainstalovat UCI konfiguraci. Umožňuje rozšíření o další moduly. Když vše selže, obsahuje recovery mód, který by měl být schopen zařízení zachránit. Aktuálně jsou k dispozici tyto moduly: DHCP, DNS, STATIC module, Sysupgrade modul, Tgz modul, Recovery modul, SslDeploy module, Service modul, Opkg modul, psaní tzv. hooks a Zabbix modul.

1.4.2 R-sync

Jedná se o nástroj určený k synchronizaci souborů a adresářů mezi zařízeními. Jeho hlavní výhody oproti příkazu *cp* jsou, že umožňuje zabezpečený přenos souborů přes ssh, dále umožňuje přenášet komprimované soubory a jednou z hlavních výhod je, že přenáší pouze rozdíly mezi jednotlivými soubory a ne vždy všechny soubory. Tento nástroj je možné používat jak na unixových distribucích, tak i na ostatních operačních systémech.

Obrázek 1.4: Logo R-sync

1.4.3 Scp

Jedná se o protokol rozvíjející jednoduché kopírovaní souborů příkazem *cp*. Umožňuje totiž zabezpečené kopírování souborů mezi dvěma zařízeními. Zkratka SCP znamená Secure Copy. Pro zabezpečení spojení proti případnému odposlouchávání slouží protokol SSH.

2 IP telefonie

VoIP je technologie, která umožňuje přenos digitalizovaného hlasu skrze počítačovou síť nebo jiné médium v těle paketu protokolů UDP/TCP IP. Proto, aby byl hlas srozumitelný a kvalitní, je důležité, aby byly zajištěny správné parametry QoS.

V rámci OpenWRT/LEDE repositáře můžeme použít hned několik softwarových balíčků, které nám umožní naplno využívat služeb IP telefonie; mezi ně patří Asterisk, Kamilo, OpenSIPS, balíčky určené primárně pro testování SIP spojení, např. sipp & sipsak, nebo jednoduchý a modulární SIP user-agent baresip.

V rámci řešení této práce padla volba na softwarové řešení B2BUA Asterisk. Hlavním důvodem této volby je jeho rozšířené používání v rámci výuky. Dalšími důvody je jeho široká uživatelská základna, obsáhlá dokumentace či jeho jednoduchá konfigurace, jejíž praktické ukázky naleznete v kapitole 4.5.

2.1 Asterisk

Historie Asterisk sahá až do roku 1999. Jeho autorem je Mark Spencer pod záštitou firmy Digium.

Obrázek 2.1: Logo Asterisk

Jedná se o OpenSource softwarové řešení, které umožňuje vytvořit ústřednu (PBX), jež může propojovat telefony, počítače a jiná zařízení běžící na různých operačních systémech, ale i hardwaru a umožňuje jim využívat a připojovat se k různým telefonním službám. Může sloužit i jako brána mezi IP telefony a veřejnou telefonní sítí (PSTN). Dnes je to již univerzální nástroj pro budování komunikačních aplikací. Asterisk už nejen že ovládá PBX systémy, ale také například VoIP gateway, systémy pro call centra, funkce pro konferenční hovory, servery pro hlasové schrány, prediktivní volič a spoustu jiných aplikací pro komunikaci v reálném čase. Podporuje několik standardů pro VoIP komunikaci, jako jsou například Session Initiation Protocol (SIP), Media Gateway Control Protocol (MGCP) a H.323. Asterisk je modulární a konfigurovatelný systém, který si každý může přizpůsobit dle svých potřeb, díky čemuž může běžet i na velmi výkonově slabých zařízeních.

Aktuální verze je Asterisk 15.1.3

2.2 Kamailio

Kamailio [4] začalo být vyvíjeno v roce 2001 výzkumnou institucí Fraunhofer Fokus sídlící v Berlíně. V té době byl název projektu SIP Express Router (zkratka SER). V roce 2005 byla vytvořena odnož tohoto projektu pod názvem OpenSER, ale z důvodu problémů s ochrannými známkami byl tento projekt přejmenován na Kamailio. Změna nastala v roce 2005, kdy došlo k postupné fúzi obou projektů a po skončení tohoto spojení již projektu zůstal název Kamailio.

Obrázek 2.2: Logo Kamailio

Jedná se o OpenSource SIP server, který je dále šířen pod GPL [1] licencí. Je možné ho používat pro vytvoření rozsáhlých platforem pro VoIP a komunikaci v reálném čase (např. WebRTC, instant messaging atd.). Má modulární architekturu. Další výhodou je možnost využití pro rozšíření bran propojující SIP a PSTN, různých PBX systémů nebo média serverů. Mezi další vlastnosti Kamailio patří asynchronní TCP, UDP a SCTP, zabezpečená komunikace prostřednictvím TLS pro VoIP (hlas, video, text), Podpora WebSocket pro WebRTC, IPv4 a IPv6, vyvažování zatížení, podpora mnoha systémů typu backend jako MySQL, Postgres, Oracle, Radius, LDAP, Redis, Cassandra, MongoDB aj.

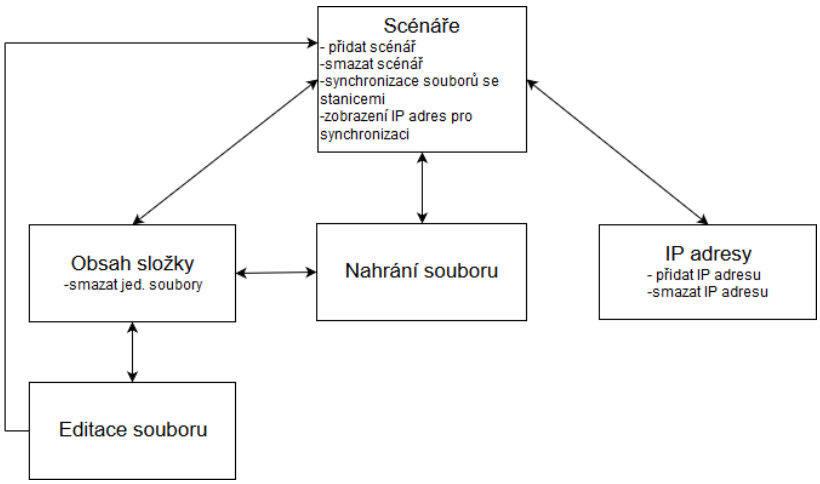
Nejnovější verze je 5.0.4.

3 Návrh automatické konfigurace IP telefonních prvků

Hlavní cílem této práce je, vytvořit modulární systém, který by byl jednoduchý na ovládání a umožnil uživateli zjednodušenou správu a rychlé nastavení parametrů ústředny Asterisk a následnou distribuci konfiguračních souborů v rámci sítě na jednotlivé stanice. V rámci konceptu byly zvažovány různé cesty řešení. Jako základ poslouží operační systém OpenWRT/LEDE, a to jak pro hlavní stanici s uživatelským rozhraním, tak i pro stanice určené ke konfiguraci, na nichž poběží SIP B2BUA Asterisk server. Jelikož OpenWRT/LEDE již pro svou vlastní uživatelsky přívětivější správu používá webové rozhraní LuCI, bude snahou toto uživatelské rozhraní rozšířit o další možnosti za použití programovacího jazyka Lua, na kterém je toto webové rozhraní postaveno tak, aby bylo dosaženo celistvého konceptu ovládání a jednoduché implementace na stávající řešení.

3.1 Koncept uživatelského rozhraní

Uživatelské rozhraní by mělo umožnit zvolit si některý z předem nastavených scénářů, popřípadě i měnit jeho parametry dle potřeby. Koncept uživatelského rozhraní byl navrhnut tak, aby uživatel, v tomto případě s velkou pravděpodobností vyučující, nemusel v co největší možné míře řešit chyby způsobené chybnou základní konfigurací ústředí Asterisk způsobené studenty a mohl se soustředit na praktické řešení zadaných úkolů se studenty. Uživatelské rozhraní by mělo mít možnost přidávat nové scénáře dle potřeby vyučování, či jejich následné mazání. Také možnost výběru a jednoznačné identifikace zařízení, se kterými mají být jednotlivá data synchronizována. Další položkou by měla být možnost editovat jednotlivé konfigurace, konkrétně konfigurační soubory tak, aby nebylo nutné je vždy kvůli editaci stáhnout, otevřít, upravit a následně opět nahrávat zpět na hlavní stanici, nebo se na ni připojovat například skrze SSH a editovat konfigurační soubory přímo na zařízení.

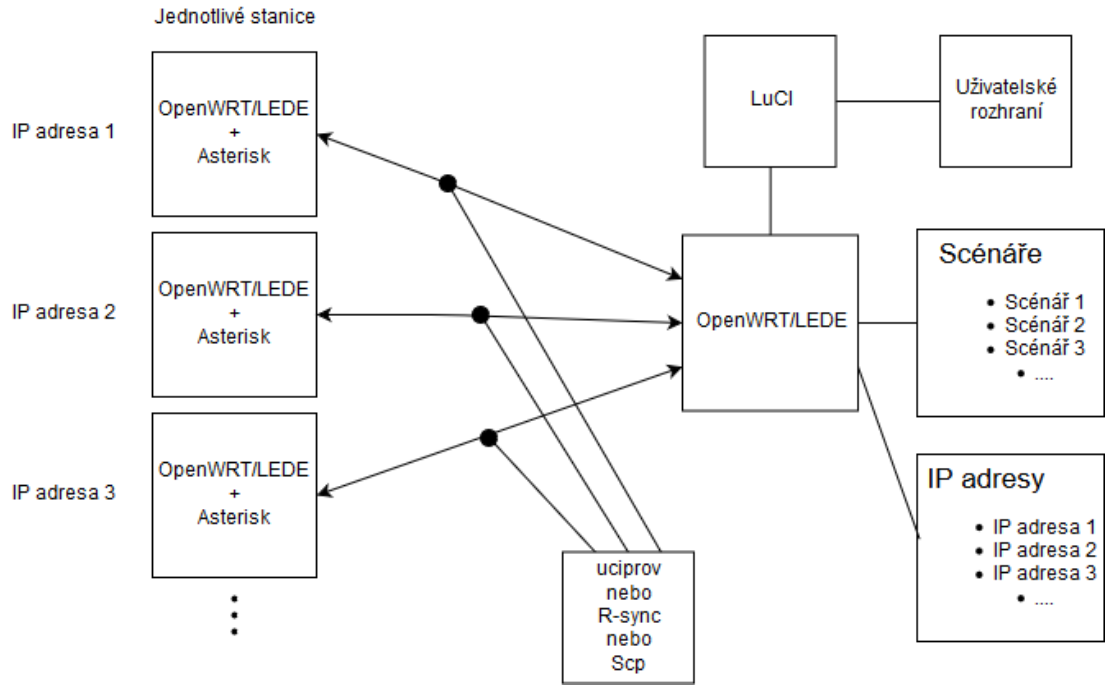


Obrázek 3.1: Grafické znázornění návrhu uživatelského prostředí

Další možností pro zvýšení uživatelského komfortu a efektivity práce by byla možnost nahrání souboru do konkrétní složky scénáře, případné mazání těchto souborů. Hlavní funkcí, kterou však je potřeba vyřešit, je synchronizace jednotlivých souborů se stanicemi v rámci sítě, tak, aby byla zajištěna jejich aktuálnost a funkčnost. Aby bylo docíleno toho, že na všech požadovaných zařízeních po synchronizaci bude načtena a aplikována stejná konfigurace SIP B2BUA Asterisk. Uživatel by také měl dostat zpětnou vazbu o tom, zda synchronizace proběhla v pořádku, nebo zda došlo k nějakému problému u konkrétního zařízení. Vyřešením všech těchto otázek by mělo dojít k vytvoření modulárního, uživatelský přívětivého prostředí, které umožní zvýšení efektivity vyučování v předmětech souvisejících s IP telefonií.

3.2 Koncept automatické konfigurace

V rámci konceptu je uvažováno několik možností řešení této problematiky. Po stisknutí tlačítka pro synchronizaci, by měly být nejprve načteny konkrétní IP adresy jednotlivých stanic, které jsou předem uživatelem zadány a uloženy v pomocném souboru. Dále je potřeba zajistit, aby se pro synchronizaci zvolily pouze konkrétní soubory, či složky, které uživatel požaduje. Tyto soubory a složky by byly uloženy na zařízení a cestu k nim by definoval uživatel v rámci uživatelského prostředí. Bylo by tedy potřeba vždy ověřit, zdali existuje požadovaná složka se soubory. Seznam těchto cest i s názvy přiřazených scénářů by byl uložen v pomocném souboru na disku. Následně by měl proběhnout přenos jednotlivých souborů, či složek na vybrané stanice, přičemž by bylo vhodné kontrolovat, zdali se na cílové stanici již nenachází stejné složky či soubory. Tak aby nedocházelo k redundanci, tudíž ke zbytečnému plýtvání zdrojů. Tato kontrola by mohla probíhat dle časového kódu souboru, rozdílu velikosti, či například názvu. To by záviselo na zvoleném řešení. Uvažované možnosti jsou tři; první je využití Uciprov, který je určen pro OpenWRT/LEDE zařízení a umožňuje automatizaci jejich konfigurace. Tento projekt je však stále ve vývoji a je k němu velice málo dokumentace. Jako další možnost se jeví R-sync, který umožňuje synchronizaci rozdílných souborů či složek mezi dvěma zařízeními. V neposlední řadě je zde Scp, který jednoduše kopíruje zvolené soubory či složky skrze zabezpečený protokol SSH do druhé zařízení.



Obrázek 3.2: Grafické znázornění konceptu

3.3 Shrnutí

Hlavní myšlenkou celého konceptu je, aby vyučují pouze zvolil vybraný scénář s předem definovanými konfiguračními soubory či jinými potřebnými soubory, IP adresy stanic a následně stiskl tlačítko pro synchronizaci. Tímto by došlo k přenosu souborů na jednotlivé stanice a následnému restartování služby tak, aby bylo zajištěno načtení nové konfigurace služby, v tomto případě služby SIP B2BUA Asterisk. Tímto by bylo dosaženo velké úspory času, jelikož ze strany vyučujícího by bylo jisté, že na všechna zařízení se přenesla stejná konfigurace, kterou si může předem překontrolovat a vyzkoušet. Není tedy potřeba případné chyby hledat v konfiguračních souborech a tímto způsobem by měla být zajištěna stoprocentní funkčnost. Odpadá možnost chyby ze strany studentů při konfiguraci. Studenti tedy již budou mít před sebou nakonfigurovaná zařízení a mohou si ověřit jejich funkčnost, prozkoumat konfiguraci a následně dle zadaných cvičení konfiguraci dále měnit a sledovat výsledky jejich snažení. Celý tento navrhovaný systém by také měl být modulární. Tak, aby existovala možnost jeho pozdějšího rozšíření o další funkce a rovněž jeho jednoduchá implementace.

4 Realizace navrhovaného systému automatické konfigurace IP telefonních prvků

V první fázi vývoje řešení automatické konfigurace bylo pracováno s možností užití UCIPROV nástroje, který umožňuje automatickou konfiguraci OpenWRT/LEDE zařízení skrze síť. Při testování funkčnosti a studiu jednotlivých modulů však bylo zjištěno, že neobsahuje zatím modul, který by se dal v tomto případě použít. Pro tento účel zkrátka nebyl tento softwarový balíček vytvořen a byla by potřeba implementace dalších balíčků. Tudíž se od této možnosti ustoupilo. Další možností, která již byla zmíněna v předchozích kapitolách, bylo použití balíčku R-sync. Tato cesta se ukázala jako správná, a tudíž byl tento balíček implementován jako primární funkce pro synchronizaci mezi stanicemi. V následujících kapitolách jsou blíže popsány jednotlivé kroky a rovněž i soubory, které dohromady tvoří funkční řešení. Jejich popis nemá nahradit programátorskou dokumentaci, ale má za cíl přiblížit použité řešení a posloužit i jako návod někomu dalšímu, kdo by chtěl toto řešení dále rozvíjet.

4.1 Instalace OpenWRT/LEDE

Z důvodu zjednodušení vývoje samotného řešení, bylo zvoleno použití virtualizačního softwaru, konkrétně softwaru s názvem VMWare, který umožňuje vytvoření virtuálního zařízení různých platforem. Předtím, než přejdeme k samotné instalaci, je potřeba připravit obraz disku s operačním systémem, v tomto konkrétním případě LEDE 17.01.4[12]. Na uvedeném odkazu naleznete repositáře s binárními obrazy určenými pro konkrétní zařízení i obrazy disku. V našem případě byl zvolen obraz disku x86-generic-combined-ext4. Po stažení souboru je potřeba jej za pomoci kompresního programu (např. Winrar) extrahovat a následně převést extrahovaný soubor formátu *img* na formát *vmdk*, který používá VMWare. Pro tento úkol se hodí například program Qemu, který je dostupný zdarma[11]. Jedná se o program spustitelný přes příkazový řádek operačního systému Windows. Pro jeho spuštění je potřeba použít příkazový řádek a přejít do složky, kde byl program extrahován a kde se nalézá spustitelný soubor qemu-img.exe. Až poté je možnost program spustit a použít. Příklad použití můžeme vidět níže.

```
qemu-img.exe convert lede-17.01.4-x86-64-combined-ext4 -O vmdk
owrt_priklad.vmdk
```

Nyní již můžeme přejít k samotnému vytvoření virtuálního stroje s operačním systémem LEDE. Pro virtualizaci byla použita konkrétní verze Mare Workstation 8. Jedná se o mnohem starší verzi, než která je nyní dostupná. Důvod je rozebrán později v této kapitole. Po spuštění programu Mare je potřeba zvolit File - New Virtual Machine. Objeví se kontextové okno s nabídkou typu instalace. V tomto případě je potřeba zvolit možnost Custom (Vlastní). V několika dalších krocích nalezneme různé možnosti nastavení virtuálního stroje. V rámci navrhovaného řešení je potřeba zvolit v části volby operačního systému, operační systém Linux, konkrétně Other Linux 2.6.x kernel 64-bit. Další položkou, kterou je potřeba nastavit je typ sítě, zde zvolit možnost Bridged, čímž docílíme přemostění spojení virtuálního stroje do sítě skrze ethernetový adaptér hostitelského zařízení [9]. Poslední důležitou možností je volba disku. V tomto bodě zvolíme již

existující disk, konkrétně soubor dříve vygenerovaný programem Qemu ve formátu *vmdk*. Všechny ostatní položky jsou již na volbě uživatele. Po dokončení vytváření virtuálního stroje a jeho spuštění by měl být inicializován start systému LEDE.

Důvodem použití starší verze software VMWare Workstation je problém, který se objevil. Jedná se o chybu, kdy při vytváření virtuálního stroje na novějším VMWare Workstation a zvolení možnosti instalace operačního systému, kdy byl vybrán konkrétní soubor s operačním systémem LEDE, došlo k nemožnosti rozšíření paměti ROM tohoto virtuálního stroje. Tudíž nebyla možná jakákoliv instalace balíčků, či ukládání souborů na tomto virtuálním stroji. Způsobem, jak se této chybě vyhnout, je právě možnost zvolení konkrétního obrazu disku jako disku virtuálního stroje. Toto řešení jsem nalezl na oficiálních stránkách OpenWRT [7]. Bohužel tato možnost se mi ve verzích VMWare Workstation novějších, než je verze 11, nepovedla nalézt. Na stejný problém narazil i vedoucí mé diplomové práce při instalaci stejného softwaru v rámci školní sítě. Tento specifický problém však může být dán využíváním virtuálních strojů, přičemž se nejedná o jediný způsob, jak provést instalaci OpenWRT/LEDE.

Po instalaci a úspěšném spuštění vytvořeného virtuálního stroje je potřeba ověřit, zdali je přístupný v rámci sítě a zda je možné se připojit na webové uživatelské rozhraní, které OpenWRT/LEDE obsahuje v rámci instalačního balíčku. Výchozí adresa uživatelského rozhraní je http://192.168.1.1; po zadání této adresy do prohlížeče by se měla zobrazit stránka s žádostí o administrátorské jméno a heslo pro správu virtuálního stroje - Obrázek 4.2 . Pokud ne, je potřeba za pomoci příkazu *ifconfig* zjistit IP adresu virtuálního stroje a tu poté zadat do webového prohlížeče. Pokud se ani po tomto kroku nezobrazí výše zmíněná stránka, je potřeba si ověřit, zdali máme správně nastavené síťové rozhraní pro virtuální stroj. Otevřeme VMWare Virtual Network Editor, kde je potřeba nastavit VMnet0 na konkrétní adaptér, který používáme pro připojení do sítě (výchozí hodnotou je automatická volba). Poté přejdeme do konzole virtuálního stroje, kde je potřeba zkontrolovat nastavení síťových rozhraní. Příkazem

```
vi /etc/config/network
```

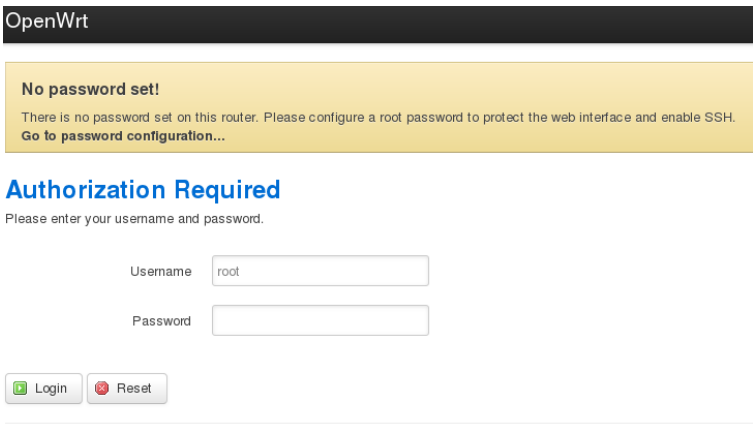
se otevře konfigurační soubor, kde jsou uvedena síťová rozhraní. Zde je potřeba nastavit u síťového rozhraní, které je používáno, možnost *dhcp* tak, aby virtuálnímu stroji byla přidělena IP adresa od DHCP serveru. Výchozí hodnotou je hodnota static. Uložení souboru provedeme stisknutím tlačítka Esc a vepsáním *:wq*. Tento krok uloží změny v souboru a soubor zavře [8].

Obrázek 4.1: Ukázka konfiguračního souboru /etc/config/network

Nyní je potřeba restartovat službu network, aby se projevily provedené změny.

```
/etc/init.d/network restart
```

Příkazem *ifconfig* ověříme, zdali byla IP adresa DHCP serverem přidělena a tuto adresu použijeme pro přístup do webového rozhraní LuCI.



Obrázek 4.2: Přihlašovací stránka OpenWRT

Výchozími přístupovými údaji jsou uživatelské jméno root a heslo admin. Po prvním přihlášení je potřeba nastavit administrátorské heslo, a tím umožnit vzdálené spojení přes SSH. Informace o tom se zobrazuje i při přihlášení viz. Obrázek 4.2 .

4.2 Instalace potřebných balíčků

Pro správnou funkcionalitu navrhovaného řešení je potřeba doinstalovat konkrétní softwarové balíčky. Na jednotlivé pracovní stanice, se kterými budou studenti pracovat, je potřeba doinstalovat jak R-sync, tak i PBX Asterisk. Avšak pro funkčnost navrhovaného uživatelského rozhraní na hlavním, řekněme učitelském stroji, stačí pouze instalace R-sync. Po instalaci všech balíčků je doporučeno provést restart příslušného zařízení, tak aby byla zajištěno, že balíčky jsou správně načteny a jsou funkční.

4.2.1 SIP B2BUA Asterisk

Balíček Asterisk je možné nainstalovat dvěma způsoby. Prvním je instalace skrze webové rozhraní LuCI, konkrétně záložky System - Software, kde je možnost instalace různých verzí a také balíčků rozšiřujících funkcionalitu. Druhou možností je instalace skrze terminál zadáním příkazů.

```
opkg update
```

```
opkg install asterisk13
```

První příkaz obnoví seznam dostupných balíčků a druhý nainstaluje balíček Asterisk ve verzi 13. V rámci zadání této diplomové práce je vytvoření několika scénářů používaných při výuce. Je tedy nutné nainstalovat i další balíčky, které rozšiřují možnosti použití Asterisk a jsou nezbytné pro zajištění funkčnosti předpřipravených scénářů.

```
opkg update
```

```
opkg install asterisk13-chan-sip
```

Rozšiřuje Asterisk o možnost využívání SIP protokolu.

```
opkg install asterisk13-res-rtp-asterisk
```

Tento balíček poskytuje podporu pro RTP, RTCP se symetrickou podporou RTP pro průchod skrze NAT.

```
opkg install asterisk13-codec-ulaw
```

Tento balíček zajišťuje podporu mezi podepsanými lineárními kodeky a ulaw kodeky.

```
opkg install asterisk13-codec-gsm
```

Podobně tento balíček zajišťuje podporu mezi podepsanými lineárními kodeky a GSM kodeky.

```
opkg install asterisk13-format-gsm
```

Jako poslední je nutný balíček, který rozšiřuje Asterisk o podporu formátu GSM.

4.2.2 **R-sync**

Instalace tohoto balíčku probíhá stejně jako u výše zmíněného Asterisk 4.2.1. Možnost první je skrze uživatelské webové rozhraní. Druhou možností je instalace za pomoci příkazů.

```
opkg update
opkg install rsync
```

4.3 Příprava stanic pro zabezpečené spojení protokolem SSH

Pro správné fungování navrhovaného řešení je potřeba dosáhnout zautomatizování zabezpečeného spojení protokolem SSH tak, aby nebyl potřeba žádný vnější zásah. Toho lze dosáhnout za pomoci ověřování prostřednictvím metody veřejného klíče. Tato metoda umožňuje bezpečné ověřování totožnosti bez potřeby zadávání hesla do konzole. Princip ověřování za pomoci klíčů je následující. Na hlavní stanici jsou vygenerovány dvě sady klíčů. Jeden je tzv. privátní klíč a druhý je veřejný. Privátní klíč je uložen ve stanici a veřejný klíč je dále distribuován na další stanice. Jelikož není možné z veřejného klíče získat klíč privátní, jedná se o velmi bezpečnou metodu ověřování identity.

V tomto případě je potřeba vygenerovat sadu klíčů na hlavní stanici, kde je uloženo uživatelské rozhraní. Vygenerování klíčů je možné za pomoci tohoto příkazu zadaného do konzole. Parametrem t se nastavuje typ generovaného klíče. Parametrem f se nastavuje cesta k souboru, kam má být uložen privátní klíč.

```
dropbearkey -t rsa -f ~/.ssh/id_dropbear
```

Privátní klíč se uloží do souboru *id_dropbear* a veřejný klíč se zobrazí v konzoli. Pokud je potřeba veřejný klíč uložit do souboru, je nutno použít příkaz, který je uveden níže. Tento

příkaz uloží veřejný klíč do souboru *public_key.txt*. Následně je možnost s tímto souborem dále pracovat.

```
dropbearkey -y -f ~/.ssh/id_dropbear > public_key.txt
```

Jakmile mám veřejný klíč k dispozici, je potřeba ho přenést na jednotlivé stanice. Nejjednodušší způsob je skrze uživatelské rozhraní jednotlivých stanic. V sekci System - Administration možnost vložení SSH klíčů. Do tohoto textového pole je potřeba vložit získaný veřejný klíč.

4.4 Realizace uživatelského rozhraní

Rozšiřování možností uživatelského rozhraní LuCI lze uskutečnit dvěma způsoby. První možností je vytváření tzv. View(template), které jsou kombinací HTML a Lua. Jsou vhodná k zobrazení některých dat, textu či obrázků. Druhou je psaní CBI modelů v programovacím jazyce Lua. Jedná se o soubory s příponou *.lua*, které popisují strukturu konfiguračního souboru Uci a výsledného HTML formuláře. Všechny soubory CBI vrací jako hodnotu objekt typu *luci.cbi.Map*. Při zpracování této diplomové práce byla většina úsilí směřována k tvorbě právě CBI modelů, které zajišťují jak funkčnost celého řešení, tak i jeho grafickou stránku.

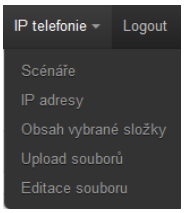
Při vytváření nových prvků LuCI rozhraní je vhodné dodržet stromovou strukturu ukládání jednotlivých prvků. Většina souborů je uložena ve výchozí složce */usr/lib/lua/luci*, kde se pak dále dělí do jednotlivých sekcí. Mezi hlavní patří složka *controller*, kde nalezneme soubory sloužící primárně pro zobrazení položek menu - Obrázek 4.4 a přiřazení jednotlivých modelů, funkcí či šablon k nim. Na obrázku 4.3 je možné vidět ukázku. LuCI používá proces, který spustí ve všech nalezených řadičích (controller) funkci index a sestaví z nich stromovou strukturu[14].

```
--- Controller
--Jednotlivé položky představují položky menu a odkazy na soubory, které se mají otevřít po stisknutí specifické volby v menu
module("luci.controller.diplomka.diplomka_controller", package.seeall)

function index()
    entry({"admin", "diplomka_cbi"}, firstchild(), "IP telefonie", 60)
    entry({"admin", "diplomka_cbi", "scenare"}, cbi("diplomka_cbi/scenare"), "Scénáře", 1)
    entry({"admin", "diplomka_cbi", "ip_adresy"}, cbi("diplomka_cbi/ip_adresy"), "IP adresy", 2)
    entry({"admin", "diplomka_cbi", "edit_scenar"}, cbi("diplomka_cbi/edit_scenar"), "Obsah vybrané složky", 3)
    entry({"admin", "diplomka_cbi", "file_upload"}, cbi("diplomka_cbi/file_upload"), "Upload souborů", 4)
    entry({"admin", "diplomka_cbi", "save_conf"}, cbi("diplomka_cbi/save_conf"), "Editace souboru", 5)
end
```

Obrázek 4.3: Ukázka souboru diplomka_controller.lua

Aby tedy došlo k zaregistrování námi požadovaných funkcí do tohoto stromu, je potřeba do funkce *index()* vložit jednotlivé prvky menu. K tomu lze použít vstupní funkci *luci.dispatcher.entry*, která má čtyři vstupní argumenty (poslední dva z nich jsou nepovinné). Prvním z nich je cesta, pod kterou bude tato větev zobrazena ve struktuře stromu. Také je možné pod touto adresou tuto větev nalézt. Druhý argument popise akci, která má být provedena při načtení této větve. Nejčastěji používané akce jsou call, template, cbi. Call vykoná požadovanou funkci. Template načte a zobrazí požadovaný HTML-template(šablonu) a v neposlední řadě CBI, která načte a zobrazí požadovaný CBI modul. Dalšími dvěma nepovinnými argumenty jsou

název, který bude zobrazován v menu, a číselná hodnota, která určí pořadí jednotlivých větví na stejné úrovni ve stromové struktuře.

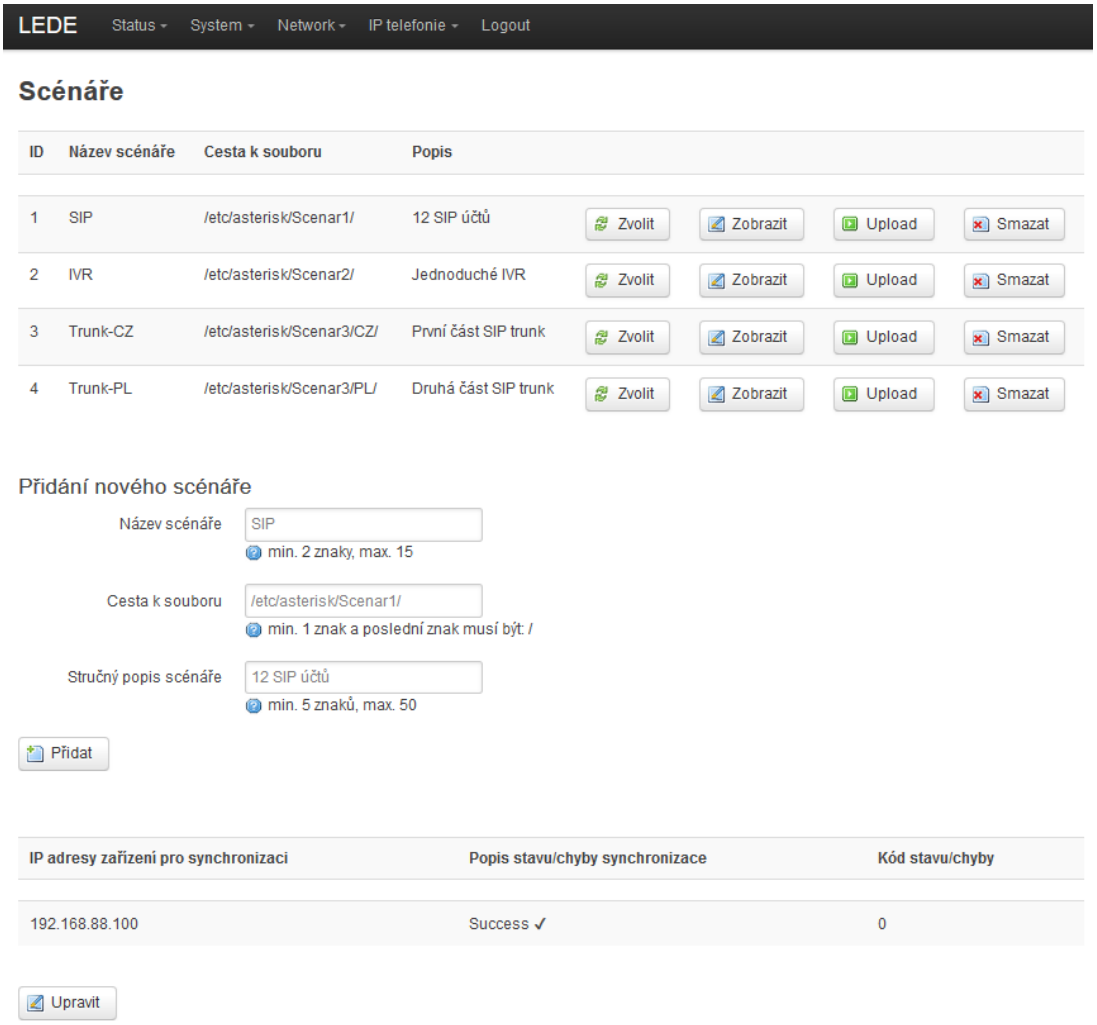


Obrázek 4.4: Ukázka výsledku funkce controller

Mezi další důležité složky se soubory patří model, kde můžeme nalézt velkou část souborů, které mají na starosti většinu hlavních funkcí. V podsložce *cbi*, která je jako výchozí pro hledání jednotlivých CBI modelů, nalezneme několik složek, které reprezentují položky menu. Zde byla vytvořena složka *diplomka_cbi*, ve které se nalézají vytvořené CBI modely zajišťující funkčnost celé realizace. Mezi další složky patří například View, ve které se nalézají jednotlivé HTML-šablony.

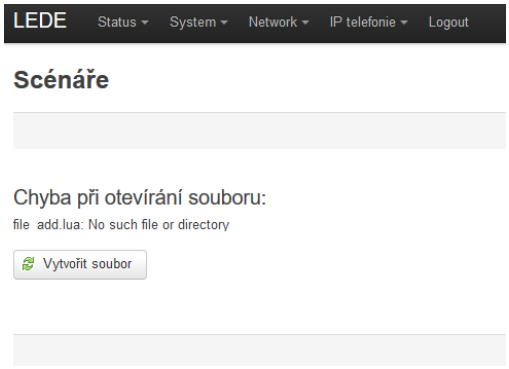
4.4.1 Hlavní okno se scénáři

Hlavním CBI modelem je *scenar.lua*. Jedná se o soubor, který vytváří hlavní uživatelskou strukturu, přes který je možné ovládat většinou prvků a provádět primární synchronizační funkci celého projektu. Struktura tohoto rozhraní je rozdělena do tří částí. První částí je zobrazení tabulky s jednotlivými scénáři, jejich údaji a tlačítka, která umožňují práci s těmito scénáři. Druhou částí je formulář s možností přidání nového scénáře. Poslední částí je zobrazení vložených IP adres, případně i stav po synchronizaci, jak je možné vidět na obrázku 4.5 .



Obrázek 4.5: Hlavní uživatelské rozhraní Scénáře

Jak již bylo zmíněno, v první části je možné vidět tabulku s jednotlivými scénáři. Každý tento scénář má své identifikační číslo ID, jednoznačný název, unikátní cestu k souboru a svůj popis. Všechny tyto parametry mají svou maximální a minimální délku, která je vyžadována, a jsou povinné. Veškerá tato data jsou uložena v pomocném souboru *file_add.lua*, který se nachází ve výchozí složce */www*. Jednotlivé položky jsou v tomto souboru odděleny znakem *;* a každý jednotlivý scénář odpovídá právě jednomu řádku v tomto souboru. Pokud tento soubor neexistuje nebo není dostupný, rozhraní se změní a uživatel je požádán o vytvoření toho souboru - Obrázek 4.6 , jelikož bez tohoto pomocného souboru není možné dále pracovat. Vytvoření tohoto pomocného souboru uživatel provede jednoduchým stisknutím příslušného tlačítka. Tím se vykoná odpovídající funkce, jež zajistí vytvoření příslušného souboru. Pokud se soubor nepovede vytvořit, zobrazí se chybová hláška.



Obrázek 4.6: Uživatelské rozhraní v případě chybějícího pomocného souboru file_add.lua

Ve zmíněné tabulce je také možno vidět několik ovládacích prvků. Jako první je tlačítko Zvolit. Stisknutí tohoto ovládacího prvku spustí funkci, jež zajistí synchronizaci souborů v závislosti na zvoleném scénáři a IP adresách nalézajících se v pomocném souboru *ip_add.lua*. Tato funkce využívá dvou tabulek, ve kterých jsou uloženy potřebné informace pro práci této funkce *file_add* a *ip_add*. První ze zmíněných obsahuje všechna data o příslušných scénářích podle jednotlivých *ID*. Při volání funkce je jako vstupní parametr důležitá proměnná *section*; jedná se o identifikační hodnotu řádku, na kterém bylo stisknuto tlačítko Zvolit. Pomocí této proměnné tedy získáme konkrétní cestu k souborům scénáře z tabulky *file_add*. Poté za pomoci cyklu *for*, který proběhne pro každou položku v tabulce *ip_add*, tedy pro každou IP adresu nalézající se v souboru *ip_add.lua*. V rámci tohoto cyklu proběhne poslání několika příkazů do konzole zařízení. Z důvodu zajištění synchronizace správné složky proběhne v rámci konzole a příkazu *cd* přechod do složky se soubory pro synchronizaci.

```
function hup.write(self, section)
    local ip_address
    local file_address = file_add[section].add
    for i in ipairs(ip_add) do
            ip_address=ip_add[i].text
            local kod = luci.sys.call("cd "..file_address..'; rsync -a  -e "ssh -y -i /root/.ssh/id_dropbear" -t * root@'..ip_address..':/etc/asterisk')
            local kod1 = luci.sys.exec('ssh -y -i /root/.ssh/id_dropbear root@'..ip_address..' "/etc/init.d/asterisk restart"')
            ip_add[i] ={text=ip_address, err=kod, text_err=rsync_err[kod]}
    end
     s:option(DummyValue, "text_err", translate("Popis stavu/chyby synchronizace"))
     s:option(DummyValue, "err", translate("Kód stavu/chyby"))
end
```

Obrázek 4.7: Ukázka funkce pro synchronizaci CBI modelu scenare.lua

Následně dojde ke spuštění služby R-sync, která vytvoří šifrované SSH spojení, kontrolu rozdílů jednotlivých souborů a složek oproti těm nacházejícím se ve složce */etc/asterisk*, která je umístěna na vzdálené pracovní stanici. V případě rozdílu dojde k přenosu těchto souborů a vygenerování návratové hodnoty v závislosti na výsledku tohoto spojení a uložení této hodnoty do proměnné *kod*. V dalším kroku proběhne znovu vytvoření zabezpečeného spojení skrze SSH a poslání příkazu do konzole na vzdálené stanici, který provede restart služby Asterisk, čím je zajištěno načtení nové konfigurace. Následně dojde k uložení proměnné *kod* do tabulky *ip_add* spolu s příslušnou hláškou odpovídající návratové hodnotě v tabulce *rsync_err* [16]. Tato data se poté zobrazí v tabulce s IP adresami, jak lze vidět na obrázku 4.5 , a u každé z nich bude možnost vidět, zdali proběhla synchronizace s touto IP adresou korektně, nebo zdali došlo k chybě, případně jaké. Bohužel zde existuje případ, kdy nastane chyba v SSH spojení, která se nepovedla vyřešit ani podchytit použitím způsobu podchycení chyby nebo použitím různých způsobů

spuštění toho příkazu tak, aby nedošlo k přesměrování na chybu - Obrázek 4.8 a návratová hodnota chyby byla identifikovatelná tak, aby ji bylo možné odchytit a vyřešit. Tato chyba vzniká při chybně zadané jedné z IP adres, pokud je jedna z IP adres mimo rozsah sítě, ve které se nachází toto zařízení.

Obrázek 4.8: Chyba Bad Gateway

Další ovládací prvky v tabulce již mají většinou navigační charakter, když směrují uživatele na jiná okna, případně pro tato okna připraví pomocné soubory. Poslední tlačítko Smazat umožňuje smazání vybraného scénáře včetně složky, která je k tomuto scénáři přiřazena. Tato funkce využívá pro práci se soubory knihovnu I/O, která se standartní součástí Lua. V ukázce - Obrázek 4.9 je možné vidět například použití funkce *io.open()*, která má dva vstupní parametry, prvním je cesta souboru a druhým poté hodnota *string*. Ta reprezentuje mód, jež má být pro otevření použit. Jednotlivé módy jsou popsány v tabulce 1.1:. Dále je možné vidět použití funkce *luci.sys.call* a to podobně jako v případě funkce pro synchornizaci - Obrázek 4.7 . V tomto případě však jako parametr funkce, která je volána, slouží funkce *rm* (remove directory entries), která umožňuje mazání souborů či složek. Tato funkce používá jako hlavní parametr cestu k souboru či složce, kterou smaže. Jako další možné parametry lze použít například *-r*, jenž je použit v tomto případě a smaže celou složku včetně jejího obsahu. Pokud vše proběhne v pořádku, návratová hodnota tohoto příkazu je 0, pokud je však návratová hodnota jakákoliv jiná, znamená to, že došlo k chybě při mazání souboru. V takovém případě scénář nebude smazán a bude vypsána chyba. Pokud je tedy návratová hodnota 0, dojde k odebrání příslušné sekce z tabulky *file_add*. Poté je pomocný soubor *file_add.lua* odstraněn a následně znovu vytvořen. Následuje zapsání do souboru *file_add.lua*. Do souboru jsou zapsána data z tabulky *file_add*, jednotlivé položky v řádku stejného *ID* jsou odděleny znakem *;* a řádky poté znakem *\n*, tedy ukončovacím znakem řádku. Pokud vše proběhne bez problému, je soubor uzavřen a stránka je obnovena tak, aby byly načteny všechny změny.

Tabulka 1.1: *Tabulka módů funkce io.open()*

Znak představující mód	Popis módu
r	Pouze pro čtení
w	Pouze pro zápis
a	Režim připojení
r+	Aktualizační mód, předchozí data zůstanou zachována
w+	Aktualizační mód, předchozí data budou vymazána
a+	Aktualizační mód, předchozí data zůstanou zachována, umožněn je pouze zápis na konec souboru
b	Speciální mód pro otevírání binárních souborů

```
function del.write(self, section)
    test_file, err = io.open("file_add.lua", "r")
    if err == nil then
        file_add = split_add(test_file)
        for m in pairs(file_add) do
            if m == section then
                err = luci.sys.call("rm -r "..file_add[m].add)
                table.remove(file_add,section)
                if err ~= 0 then
                    return error(12, err)
                end
            end
        end
       os.remove("file_add.lua")
       file, err = io.open("file_add.lua", "a")
       if err == nil then
            for m in pairs(file_add) do
                if m==1 then file:write(file_add[m].add..";"..file_add[m].text..";"..file_add[m].popis)
                else
                   file:write("\n"..file_add[m].add..";"..file_add[m].text..";"..file_add[m].popis)
                end
            end
            file:close()
            luci.http.redirect(luci.dispatcher.build_url("admin/diplomka_cbi/scenare"))
        else
            error(3,err)
        end
    else
         error(3,err)
    end
end
```

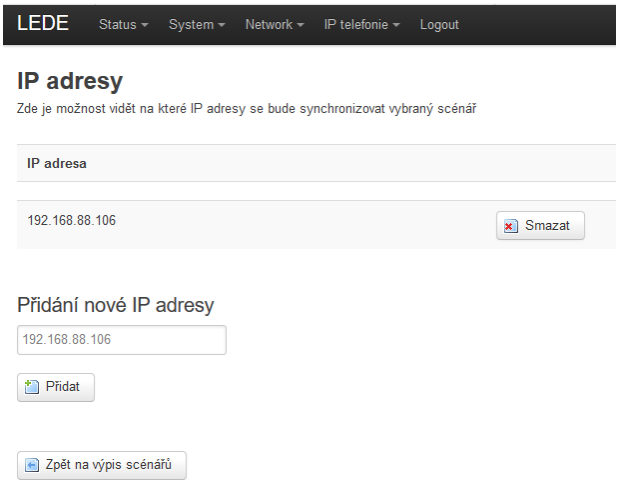
Obrázek 4.9: Ukázka funkce pro smazání scénáře

Další část toho uživatelského rozhraní se skládá z několika textových polí, do kterých uživatel vepisuje údaje o scénáři, který chce nově přidat do seznamu. Všechny údaje jsou povinné a jejich parametry kontrolovány dle požadavků, které jsou popsány pod každým textovým polem. Po stisknutí tlačítka Přidat je provedena kontrola zadaných vstupních hodnot; pokud některá z hodnot neodpovídá požadovaným parametrům, je vypsána chyba. V opačném případě dojde k otevření pomocného souboru *file_add.lua*, do kterého se na konec tohoto souboru zapíšou zadané hodnoty a proběhne znovu načtení této stránky.

V neposlední řadě se zde také nachází sekce zobrazující tabulku hodnot IP adresy načtených z pomocného souboru *ip_add.lua*. Jak již bylo zmíněno, v případě stisknutí tlačítka Zvolit se v této tabulce rovněž objeví stav synchronizace pro jednotlivé IP adresy - Obrázek 4.5 . Poslední položkou v této sekci je tlačítko Upravit, které přesměruje uživatele na CBI modul pro úpravu IP adres. Pokud pomocný soubor *ip_add.lua* neexistuje nebo není dostupný, uživateli se zobrazí okno podobné tomu, které se zobrazí v případě chybějícího pomocného souboru *file_add.lua* - Obrázek 4.6 . V tomto případě však po stisknutí příslušného tlačítka dojde k vytvoření tohoto pomocného souboru.

4.4.2 Správa IP adres

CBI modul obsahující uživatelské rozhraní a veškeré funkce pro práci s IP adresami jednotlivých stanic se nachází v souboru s názvem *ip_adresy.lua*. Toto rozhraní umožňuje zobrazení, mazání a přidávání jednotlivých IP adres dle požadavků uživatele.



Obrázek 4.10: Uživatelské rozhraní pro správu IP adres

Nacházejí se zde dvě sekce. První je tabulka s jednotlivými IP adresami, přičemž v druhém sloupci u každé IP adresy se nachází tlačítko pro možnost smazání této IP adresy - Obrázek 4.11 . Tabulka je plněna daty z pomocného souboru *ip_add.lua*, který se nachází ve složce */www*. Pokud by nastala situace, že tento soubor by nebyl dostupný, uživatelské rozhraní se změní podobně jako v případě rozhraní se scénáři - Obrázek 4.6 . Jen s tím rozdílem, že zde by tlačítko vytvořilo soubor *ip_add.lua* a došlo by k přesměrování na CBI modul *ip_adresy.lua*. V tomto pomocném souboru jsou jednotlivé IP adresy oddělený znakem , . Funkce pro smazání pracuje tím způsobem, že načte všechna uložená data z pomocného souboru *ip_add.lua* skrze funkci *split(),* která vrací načtená data do proměnné typu table s názvem *ip_add*. Nyní je za pomoci cyklu *for* celá tabula *ip_add* prohledána, přičemž data z tabulky jsou porovnávána dle ID; jakmile cyklus narazí na shodné ID s hodnotou *section*, dojde k odstranění tohoto záznamu ze zmíněné tabulky. Hodnota *section* odpovídá hodnotě řádku, ve kterém bylo stisknuto tlačítko Smazat. Následně je smazán pomocný soubor *ip_add.lua* a hned v následujícím kroku je tento pomocný soubor opět vytvořen za pomoci funkce *io.open()*, který má dva vstupní parametry, prvním je cesta k souboru včetně názvu k souboru a druhým je mód, ve kterém má být soubor otevřen. Jednotlivé možností jsou popsány v tabulce 1.1. Jak již bylo uvedeno v kapitole 4.4.1, funkce je součástí knihovny I/O, jež je součástí standartní knihovny v rámci Lua. Pokud při otevírání souboru dojde k chybě, funkce vrátí jako první parametr *nil* a jako druhý chybovou hlášku datového typu *string*. Následuje cyklus *for*, který provede zápis dat z tabulky *ip_add* do znovu vytvořeného pomocného souboru *ip_add.lua*. Poté je otevřený soubor zavřen a stránka obnovena tak, aby došlo znovu k načtení dat v tabulce s IP adresami. Zbývající část z ukázky kódu jsou kontroly různých chybových hlášení a následné zobrazení těchto chybových hlášení.

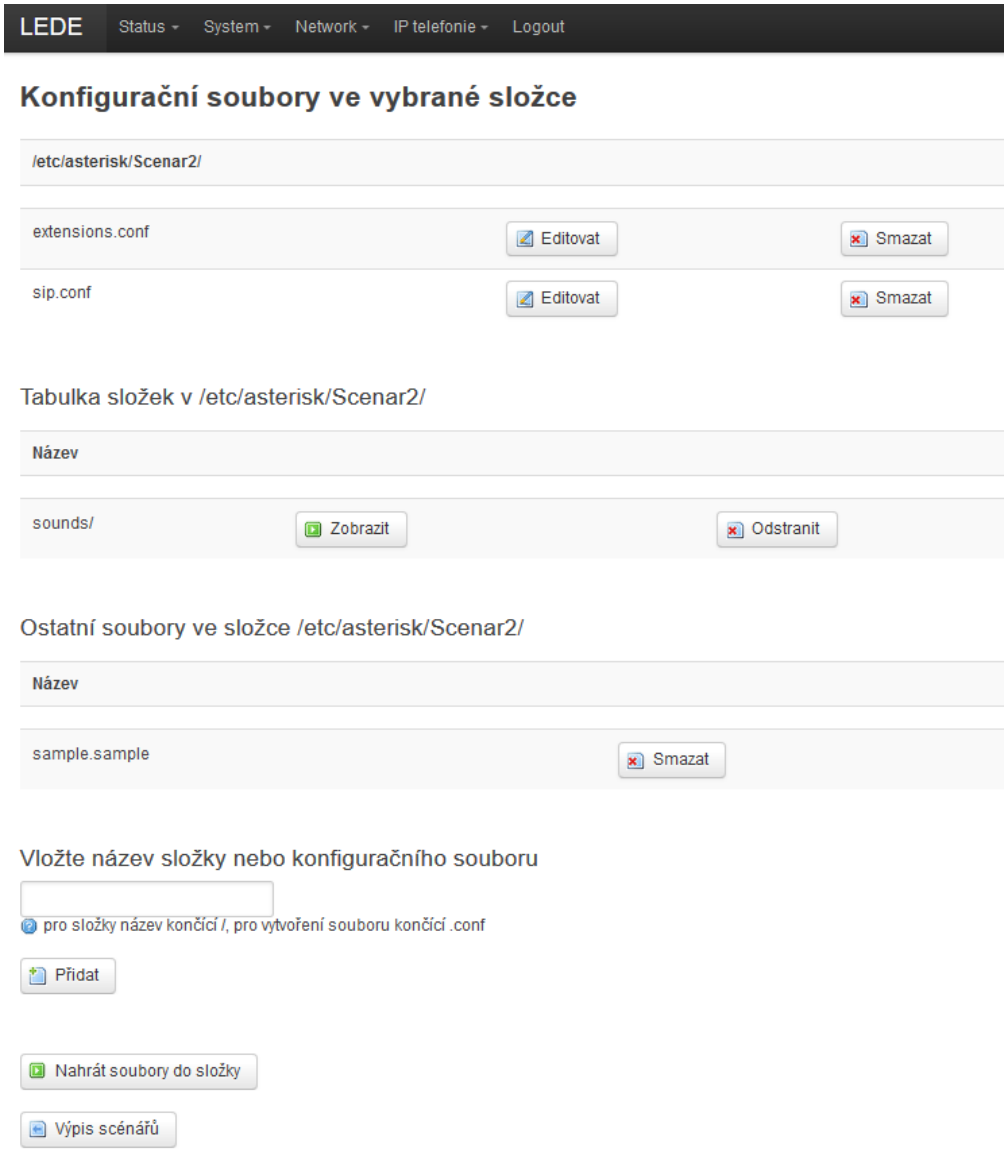
```
function del.write(self, section)
    test_file, err = io.open("ip_add.lua", "r")
    if err == nil then
        ip_add = split(io.open("ip_add.lua", "r"):read())
        for m in pairs(ip_add) do
            if m == section then table.remove(ip_add,section) end
        end
       os.remove("ip_add.lua")
       if err == nil then
           file, err = io.open("ip_add.lua", "a")
           if err == nil then
                for m in pairs(ip_add) do
                    if m==1 then file:write(ip_add[m].text)
                    else
                        file:write(","..ip_add[m].text)
                    end
                end
                file:close()
                luci.http.redirect(luci.dispatcher.build_url("admin/diplomka_cbi/ip_adresy"))
            else
                err = g:option(DummyValue, er[3])
                err.value = er[3]
            end
        else
            err = g:option(DummyValue, er[4])
            err.value = er[4]
        end
    else
        err = g:option(DummyValue, er[3])
        err.value = er[3]
    end
end
```

Obrázek 4.11: Ukázka funkce pro smazání IP adresy z pomocného souboru ip_add.lua

V druhé sekci se nachází několik prvků. Úkolem této sekce je umožnit uživateli vložit novou IP adresu. Po stisknutí tlačítka Přidat dojde ke kontrole zadaných vstupních dat, přičemž povolen je pouze standartní formát IP adresy verze 4. Důvodem je, že v rámci vyučování předmětů souvisejících s IP telefonií se používají IP adresy verze 4. V rámci dalšího vývoje a s ohledem na rozšiřování používání IP adres verze 6 je možnost úpravy konfiguračních souborů tohoto projektu a přidání podpory IP verze 6.

4.4.3 Obsah vybrané složky

Na toto rozhraní je uživatel přesměrován po stisknutí tlačítka Zobrazit, umístěném na hlavním uživatelském rozhraní Scénáře 4.4.1, nebo skrze položku v menu IP telefonie - Obsah vybrané složky. Pokud vybraná složka neexistuje, bude vytvořena. Tento CBI modul se nachází v souboru s názvem *edit_scenar.lua*. Toto uživatelské rozhraní funguje jako určitá křižovatka, na které je možno vidět obsah vybrané složky a lze se rozhodnout, zda jednotlivé soubory editovat, mazat či nahrát soubor nový. V nadpisu tabulky se zobrazuje úplná cesta ke složce, jejíž obsah je vypsán. Tato hodnota je načtena z pomocného souboru *scenar_edit.lua*. Pokud tento soubor není dostupný, objeví se okno s požadavkem, aby uživatel nejprve skrze tlačítko Zobrazit v hlavním okně se scénáři 4.4.1 vybral složku, jejíž obsah chce zobrazit. V tomto rozhraní je možné vidět několik tabulek v závislosti na typech souborů, které se nacházejí ve vybrané složce. Na obrázku 4.12 je možné vidět všechny tabulky. Důvodem tohoto rozdělení je zajištění správné funkčnosti navrhovaného řešení tak, aby bylo dosaženo zobrazení všech dat ve složce, ale rovněž, aby se předešlo zbytečným chybám. Například pokusu o editaci složky. Z toho důvodu je pouze u souborů s příponou *.conf* možno vidět tlačítko pro editaci, kterou provedete uložením názvu vybraného souboru do pomocného souboru *conf_edit.lua* a následně přesměrováním na CBI modul pro zobrazení obsahu vybraného souboru.



Obrázek 4.12: Uživatelské rozhraní CBI modulu edit_scenar.lua

Další tabulka obsahuje výpis složek, které se nacházejí ve vybrané složce. V druhém sloupci u názvů těchto složek je možné vidět tlačítko Zobrazit, kterým provedete zápis názvu vybrané složky do pomocného souboru *scenar_edit.lua* a opakované načtení tohoto CBI modulu tak, aby byl zobrazen obsah vybrané složky. Při vývoji tohoto modulu bylo potřeba vyřešit, jakým způsobem rozlišit jednotlivé soubory a složky. Bohužel samotný příkaz *ls* nevrací hodnoty dle typu souboru, ale pouze názvy všech položek jako hodnoty. Bylo tedy potřeba složky nějakým způsobem odlišit od jiných souborů. Řešením bylo použití parametru *-p*, kterým lze docílit toho, že při výpisu obsahu souboru za pomoci funkce *ls* budou názvy složek končit znakem /. V ukázce vybrané pasáže CBI modulu *edit_scenar.lua* - Obrázek 4.13 je možné vidět získání názvů všech položek nalézajících se ve složce a následné rozdělení získaných dat dle předem nadefinovaných parametrů a jejich uložení do jednotlivých tabulek. Prvním z kontrolovaných parametrů je, zdali

některý z názvů těchto dat nekončí koncovkou *.conf*, která představuje koncovku konfiguračních souborů; pokud ano tento název je uložen do proměnné *files_tab* typu tabulka. Druhou kontrolou je to, zdali název nekončí zmíněným znakem /, jenž reprezentuje složky. Pokud je tato podmínka splněna, název je uložen do proměnné typu tabulka s názvem *folder_tab*. Všechny ostatní názvy jsou uloženy do tabulky s názvem *else_tab*.

```
local text_1, err= io.popen("ls -p "..add_tab[1].text) --vypsaní obsahu složky
if err == nil then
        local j = 1
        for name in text_1:lines() do
            if name:match("(.+).conf") then --zobrazení pouze konfiguračních souborů s koncovkou conf
                files_tab[j] = {text = name}
                j=j+1
            else
                if name:match("(.+)/") then --vybraní složek (končí symbolem /)
                    folder_tab[j] = {text = name}
                    j=j+1
                else
                    else_tab[j] = {text = name} --zbývající položky ve složce
                    j=j+1
                end
            end
        end
```

Obrázek 4.13: Ukázka vybrané pasáže kódu v CBI modulu edit_scenar.lua

Právě poslední tabulka v pořadí představuje ostatní soubory, které mají jinou příponu nežli *.conf* a nejedná se o složky. U těchto položek je možné soubory pouze smazat.

V další sekci je možné vytvořit konfigurační soubor nebo složku. Pokud chce uživatel vytvořit složku, je potřeba, aby na konci názvu byl znak /, pokud chce vytvořit konfigurační soubor je potřeba k názvu přidat příponu *.conf*; všechny tyto parametry jsou hlídány a není možné vytvoření souboru jiného typu. Také je zde implementována funkce, která kontroluje, zdali název, který uživatel zadal, se již nenachází v této složce, a pokud ano, uživatele o tom vyrozumí a neumožní mu vytvoření takovéhoto souboru, či složky.

Následuje již jen sekce s tlačítky pro navigaci mezi jednotlivými moduly.

4.4.4 Upload souborů

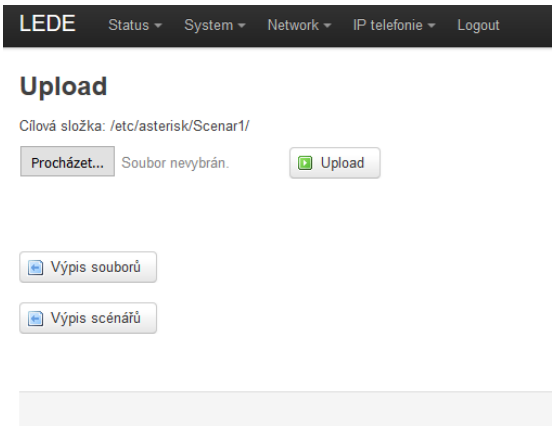
Tento CBI modul je uložen jako *file_upload.lua*. Tento modul pro svou správnou funkčnost potřebuje dva další soubory nacházející se v *../lua/luci/view/cbi*. Jedná se o soubory *other_upload.htm* - Obrázek 4.14 a *other_dvalue.htm*, které byly převzaty i s několika funkcemi z projektu wifidog OpenWrt luci页面 [13]. Jedná se o šablony, které jsou při zobrazení rozhraní načteny jako součást uživatelského rozhraní a jsou nezbytné pro hlavní funkci tohoto modulu.

```
<% cbi valueheader%>
    <input class="cbi-input-file" style="width: 400px" type="file" id="ulfile" name="ulfile" />
    <input type="submit" class="cbi-button cbi-input-apply" name="upload" value="<% Upload%>" />
<% cbi valuefooter%>
```

Obrázek 4.14: Obsah souboru other_upload.htm

Hlavní funkcí je umožnit uživateli jednoduché nahrání souboru do předem vybrané složky. O kterou složku se jedná, je zobrazeno pod názvem Upload - Obrázek 4.15 . Tento údaj je získán z pomocného souboru *scenar_edit.lua*. Pokud tento soubor není dostupný, zobrazí se hlášky s požadavkem na zvolení složky, do které se má soubor nahrát za pomocí tlačítka

nacházejícího se na okně se scénáři 4.4.1. Po stisknutí tlačítka Procházet dojde k otevření standardního kontextového okna pro procházení souborů. Toto okno umožní uživateli nalézt a vybrat soubor na svém lokálním zařízení, který chce nahrát na vzdálenou stanici. Po zvolení již stačí jen provést nahrání souboru stiskem tlačítka Upload. Po tomto kroku se zobrazí hláška o úspěšném nahrání souboru, nebo chybová hláška s informací o vzniklé chybě.



Obrázek 4.15: Uživatelské rozhraní CBI modulu file_upload.lua

Poté se na tomto rozhraní nacházejí již jen tlačítka pro navigaci mezi jednotlivými CBI moduly, respektive uživatelskými rozhraními.

Jak jsou jednotlivé sekce vytvořeny, je možné vidět na obrázku 4.16 . Nejprve je vytvořen formulář třídy SimpleForm, jenž je použit pro zobrazení statického obsahu. U tohoto formuláře jsou zakázány tlačítka Reset a Submit, která jsou součástí tohoto formuláře, avšak pro funkci tohoto okna nejsou potřeba. Následuje přiřazení první sekce SimpleSection, která je vlastně podsekcí SimpleForm a právě k SimpleForm je přiřazena skrze *m:section*, přičemž proměnná m reprezentuje právě vytvořený formulář. Posléze jsou do této sekce vloženy potřebné prvky skrze parametr *option*. Prvním z nich je FileUpload, představující třídu s již přednastavenými parametry pro upload souborů, hlavně otevření okna pro výběr požadovaného souboru a jenž je v tomto konkrétním případě vykreslen dle nastavení parametrů šablony *other_upload.htm* - Obrázek 4.14 . Následuje další šablona představující následný text, který zobrazí výsledek požadavku o nahrání souboru. Oba tyto prvky načítají jejich šablonu skrze parametr t*emplate*. Nyní již následuje další sekce bez názvu, jenž slouží pro oddělení navigačních tlačítek od sekce pro nahrávání souborů. V této sekci se nacházejí jednotlivá tlačítka pro navigaci mezi okny. V ukázce je možné vidět pouze jejich statické vytvoření, funkce těchto tlačítek nejsou v ukázce vidět. Stejně tak není možné vidět návratovou hodnotu return m, která je na samém konci souboru a bez které by nedošlo k vykreslení tohoto okna.

```
---Formulář, který zobrazuje informace o cílové složce a umožňuje nahrání souboru
m = SimpleForm("upload", translate("Upload"))
m.reset = false
m.submit = false

sul = m:section(SimpleSection, "", "Cílová složka: "..add_tab[1].text)
fu = sul:option(FileUpload, "")
fu.template = "cbi/other_upload"
um = sul:option(DummyValue, "", nil)
um.template = "cbi/other_dvalue"

--Sekce s navigačními tlačítky
g=m:section(SimpleSection,"")
button_1 = g:option(Button, "_button_1")
button_1.inputstyle = "link"
button_1.inputtitle = translate("Výpis souborů")
```

Obrázek 4.16: Ukázka CBI modulu file_upload.lua

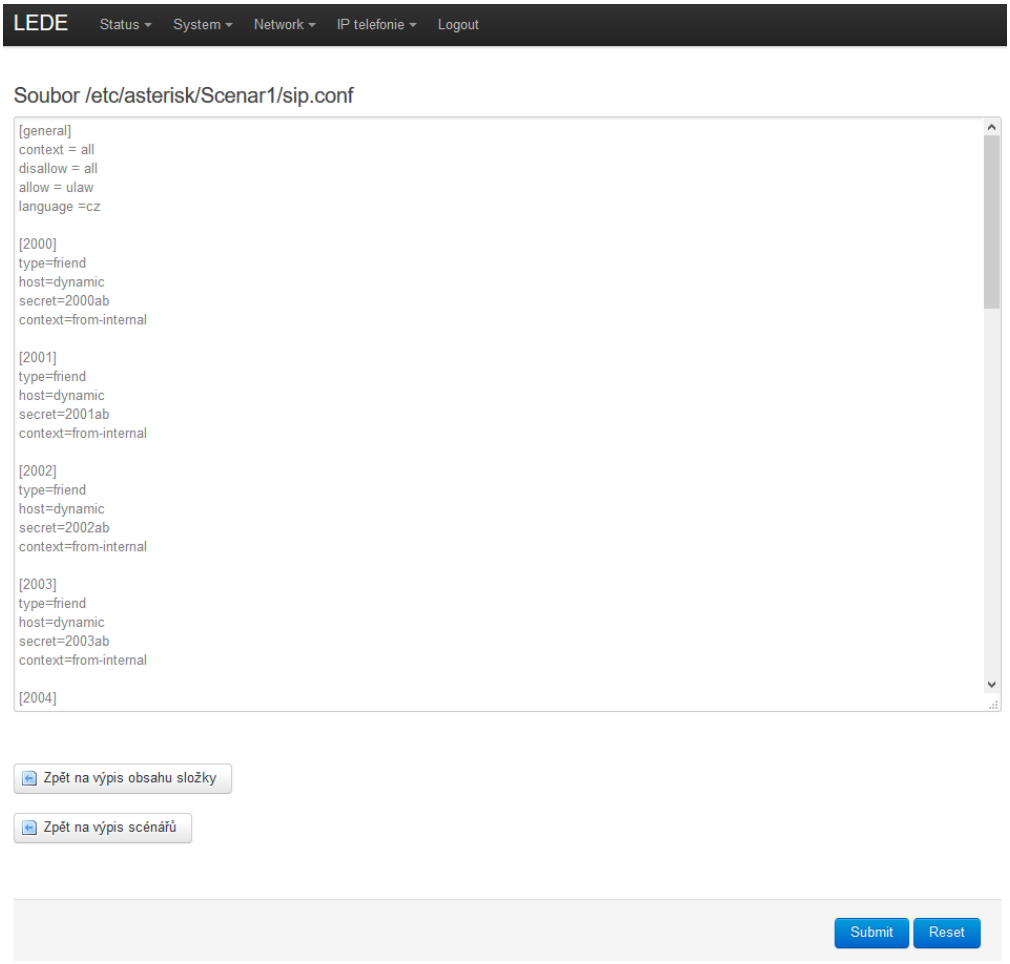
4.4.5 Editace souboru

Tento modul je uložen v souboru *save_conf.lua*; pro svou správnou funkci vyžaduje pomocný soubor *conf_edit.lua* a také *scenare_edit.lua*. Pokud jeden z nich neexistuje, zobrazí se hláška s požadavkem, aby uživatel nejprve přešel buď na hlavní okno se scénáři 4.4.1, nebo na okno s obsahem vybrané složky 4.4.3 v závislosti na tom, který ze souborů chybí. Z těchto dvou pomocných souborů je poté složena úplná cesta k souboru, který chce uživatel editovat. Tato hodnota je také zobrazena jako nadpis tohoto uživatelského rozhraní. Pod tímto nadpisem se nachází textové pole, které je nastaveno na délku 30 řádků a do něhož je načten obsah vybraného souboru. Jak je možné vidět na obrázku 4.17 , toto textové pole je načteno do samostatné sekce tohoto formuláře. Pro zobrazení byla použita třída SimpleSection. Samotné textové pole se pak nachází až v podtřídě záložky s názvem *files* a jedná se o typ TextValue, jenž je nastaven na délku 30 řádků parametrem *rows*. Pro získání a zobrazení dat slouží funkce *text_field.cfgvalue()*; *text_field* je název proměnné odkazující na textové pole a funkce *cfgvalue()* je funkce, která vrací hodnotu typu *uci* zvoleného objektu z jeho konfiguračního souboru. V tomto konkrétním případě vrací načtený obsah souboru v bufferu za pomoci modulu *nixio.fs* [15] určeného pro práci se souborovými systémy. Tento modul je načten příkazem *require* na začátku konfiguračního souboru a uložen právě do proměnné *fs*. Jelikož úkolem této funkce je načíst data ze souboru, je použita funkce *readfile()*, která může mít dva parametry. Prvním je cesta k souboru a druhým je limit maxima bajtů, které mohou být načteny. Druhý parametr je volitelný.

```lua
local add_tab = read_add(address)
local files_tab= read_add(file_add)

r=SimpleForm("")
string = add_tab[1].text..files_tab[1].text
s = r:section(SimpleSection, "Soubor "..string)
s:tab("files")
text_field = s:taboption("files", TextValue, "values")
text_field.rows = 30

---Funkce text_field.cfgvalue() provede načtení požadovaného souboru
--@return fs.readfile(string)
--@return ""
function text_field.cfgvalue()
    return fs.readfile(string) or ""
end

---Funkce text_field.write provede uložení změn do souboru
--@param self
--@param section
--@param value
function text_field.write(self, section, value)
    if value then
        fs.writefile(string, value:gsub("\r\n", "\n"):gsub("\r", "\n"))
    end
end
```

Obrázek 4.17: Ukázka ze souboru save_conf.lua

V tomto textovém poli je tedy možné provádět editaci načteného souboru. K uložení provedených změn dojde po stisknutí tlačítka Submit, které spustí funkci *text_field.write()*. Funkce uloží změny v souboru, avšak pouze pokud je v souboru alespoň nějaký znak (hodnota *value* není *nil*). Samotné zapsání se provede za pomoci již zmíněného modulu *nixio.fs* [15], avšak v tomto případě za pomoci funkce *writefile()*, která na vstupu vyžaduje dva parametry; prvním je cesta k souboru a druhým jsou požadovaná data, která se mají zapsat. U druhého parametru zmíněné funkce na obrázku 4.17 dochází k použití funkce z knihovny určené pro práci s daty typu *string*. Konkrétně o funkci *gsub()*, jejímž úkolem je nahrazení určitých znaků nalezených v textu jinými. V tomto případě se jedná o ošetření možného otevírání souborů s jiným formátováním a převedení různých typů zakončení řádků na typ *\n*, aby bylo dosaženo správného zobrazení obsahu souboru. Poslední položkou pro práci s načteným souborem je tlačítko Reset, které zruší provedené změny.



Obrázek 4.18: Uživatelské rozhraní CBI modulu save_conf.lua

4.5 Konfigurace jednotlivých scénářů

V rámci zadání této práce je jedním z bodů vytvoření několika modelových scénářů, které se používají při výuce související s IP telefonní na VŠB-TUO. V této kapitole budou představeny jednotlivé konfigurace. V kapitole 4.6 je pak popsáno testování těchto scénářů.

4.5.1 SIP

Úkolem v tomto scénáři je vytvoření dvanácti SIP účtů v rozsahu 2000 až 2012 s nastavenými hesly odpovídajícím hodnotám 2000ab až 2012ab. Cílem této konfigurace je možnost připojit se na kterýkoliv z výše zmíněných SIP účtů a také se na kterýkoliv dovolat použitím SIP protokolu. Pro konfiguraci tohoto scénáře je potřeba nakonfigurovat dva konfigurační soubory *sip.conf* a *extension.conf*.

4.5.1.1 ***sip.conf***

Jedná se o soubor, který obsahuje konfigurace účtů (přípojek), které využívají SIP protokol na této ústředně. V sekci General jsou nejprve nastaveny globální hodnoty.

```
[general]
context = all
disallow = all
allow = ulaw
language =cz
```

Následně se již jedná o konfiguraci jednotlivých přípojek. Jako příklad je zde uvedena pouze jedna přípojka, ale ostatní se liší pouze jiným ID a heslem. V hranatých závorkách je uvedeno ID přípojky, následuje *type=friend*; tato volba zajistí umožnění funkcí volat i přijímat hovory. Další položkou je *host=dynamic*; tato volba umožní přihlášení této přípojky z kterékoliv IP adresy. Parametr *secret=2000ab* nastavuje heslo, za pomoci kterého se bude zařízení připojovat na ústřednu. Posledním parametrem je c*ontext=from-internal*, jenž odkazuje na kontext v souboru *extension.conf*, který říká, co se má v případě hovoru provést.

```
[2000]
type=friend
host=dynamic
secret=2000ab
context=from-internal
```

4.5.1.2 ***extension.conf***

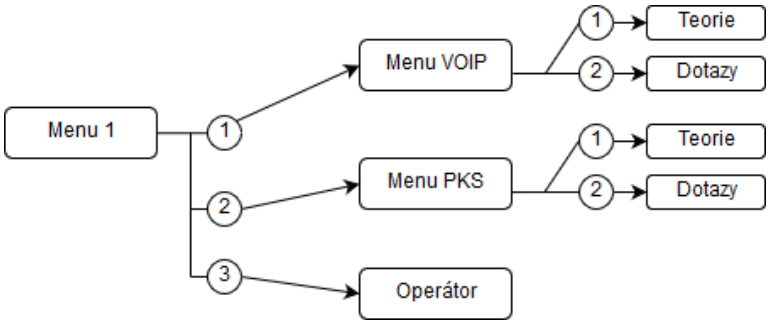
Tento konfigurační soubor obsahuje dialplan ústředny. Kontexty, které jsme uváděli v konfiguračním souboru *sip.conf*, obsahují dialplan pro jednotlivý kontext. Jelikož v *sip.conf* jsme nadefinovali jako kontext *from-internal*, musíme stejný název uvést i zde. A pak již jen parametr exten, jenž reprezentuje určité pravidlo, jehož prvním parametrem je telefonní číslo, druhým priorita a třetím funkce s parametry. V našem případě funkce Dial s parametrem typ spojení/identifikátor. Níže je zobrazena jen krátká ukázka konfiguračního souboru *extension.conf* pro tento scénář. Podobné pravidlo je potřeba nastavit pro každou přípojku zvlášť.

```
[from-internal]
exten = 2000,1,Dial(SIP/2000)
```

4.5.2 **IVR**

Cílem tohoto scénáře je vytvořit jednoduché IVR, které bude obsahovat alespoň dvě menu, základní rozhodovací strom a volání na jednotlivá čísla. Systém IVR je hlasový systém, který umožňuje provádění různých úkonů na základě předem definovaných parametrů a vstupních údajů volajícího. Ve firmách může například sloužit ke správnému přesměrování volajícího na oddělení, které potřebuje. Před samotnou konfigurací bylo potřeba vytvořit základní rozhodovací strom - Obrázek 4.19 . Podle tohoto rozhodovacího stromu pak byla vytvořena jednotlivá menu a byly nahrány zvukové soubory, které se přehrají v jednotlivých úsecích tohoto IVR. Zvukové

soubory jsou uloženy v podsložce *sounds*, která se nachází ve stejné složce jako konfigurační soubory.



Obrázek 4.19: Schéma návrhu IVR

4.5.2.1 ***sip.conf***

Zde lze vidět ukázku části konfiguračního souboru *sip.conf* určenou pro tento scénář. V první části je možné vidět globální stavení (sekce *general*). Prvním parametrem je nastavení příslušného kontextu, druhým je *bindaddr*, který určuje na kterých IP adresách má poslouchat. Nastavení 0.0.0.0 odpovídá tomu, že bude poslouchat všechny IP adresy na všech rozhraních. Jedná se také o výchozí hodnotu. Dalším parametrem je *host=dynamic*, což umožní přihlášení z jakékoliv IP adresy. *Type = friend* umožňuje volat i přijímat příchozí volání.

```
[general]
context = ivr
bindaddr=0.0.0.0
host = dynamic
type=friend
```

V další části již je ukázka nadefinování jedné z přípojek, jejichž parametry jsou prakticky shodné s nastavením v prvním scénáři 4.5.1.1. Liší se pouze v kontextu, ke kterému je přiřazen.

```
[2000]
type=friend
host=dynamic
secret=2000ab
context=ivr
```

4.5.2.2 ***extension.conf***

Zde probíhá hlavní nastavení celého IVR systému. Zde se vytvářejí dialplan pro jednotlivá menu a položky v menu. V této kapitole je možné vidět kousek z konfiguračního souboru *extension.conf*, který je určen pro tento scénář. Jedná se pouze o první část menu, o hlavní menu, do kterého bude uživatel přesměrován po zavolání na číslo 3000.

```
[ivr]
```

Název kontextu.

```
exten => 3000,1,Answer()
```

Při volání čísla 3000 bude s prioritou jedna vykonána funkce Answer(), tudíž bude hovor přijat.

```
exten=> 3000,2,Background(/etc/asterisk/sounds/hlavni_menu)
```

Následně, s prioritou dvě, bude přehrána nahrávka s názvem *hlavni_menu* za pomoci funkce Background(), přičemž jako vstupní parametr je uvedena celá cesta k souboru.

```
exten=> 3000,3,Wait(2)
```

Vyčkávání po dobu dvou sekund na volajícího reakci.

```
exten=> 3000,4,Goto(2)
```

Návrat na funkci s prioritou 2, tudíž bude znovu přehrána nahrávka *hlavni_menu*.

```
exten => 1,1,Goto(VOIP,4000,1)
```

Pokud bylo na vstupním zařízení stisknuto číslo 1, dojde k přesměrování do kontextu VOIP s volacím číslem 4000 na funkci s prioritou jedna.

```
exten => 2,1,Goto(PKS,5000,1)
```

Pokud bylo na vstupním zařízení stisknuto číslo 2, dojde k přesměrování do kontextu PKS s volacím číslem 5000 na funkci s prioritou jedna.

```
exten => 3,1,Dial(SIP/3001)
```

Pokud bylo na vstupním zařízení stisknuto číslo 3, dojde k požadavku o vytvoření SIP spojení na číslo 3001.

```
exten => i,1,Goto(ivr,3000,2)
```

Pokud bylo na vstupním zařízení zadáno cokoliv jiného, dojde k přesměrování zpět na funkci s prioritou 2.

Další položky menu jsou velice podobné výše popsané konfiguraci a liší se pouze názvem, případně použitými funkcemi.

4.5.3 SIP Trunk

Cílem tohoto scénáře je vytvořit spojení mezi dvěma stanicemi, na kterých běží Asterisk server, tak aby bylo možné zavolat mezi stanicemi. Tento scénář má simulovat reálnou situaci, kdy nějaká menší firma má dvě různé pobočky ve dvou různých zemích a potřebuje, aby byla umožněna komunikace z jedné pobočky do druhé.

Konfigurace tohoto scénáře je potřeba provést pro dvě různé stanice, na kterých běží Asterisk. Jejich konfigurační soubory se mírně liší. V kapitole 4.5.3.1 jsou konfigurace pro tyto dvě ústředny vedle sebe tak, aby bylo možné názorně vidět rozdíly mezi nimi. V levém sloupci se nachází konfigurace pro stranu CZ a v pravém pak pro PL.

4.5.3.1 ***sip.conf***

Tento konfigurační soubor je nakonfigurován podobně jako v případě konfigurace SIP spojení 4.5.1.1. V tomto případě je na straně CZ provedena konfigurace účtu alice a na straně PL bob. Rozdíl je v parametru *context*, tedy hodnotě kontextu, do které je tento účet přiřazen, *callerid*, kde je prvním parametrem název účtu a druhým je číslo, pod kterým lze tento účet volat. Dalším parametrem je *qualify=yes*, jenž umožní kontrolu stavu. V případě, že odpovídá správně, po vložení příkazu *sip show peers* se zobrazí status OK. Tato možnost není nutná a konfigurace bude plnit svou funkci i bez tohoto parametru, avšak tento parametr umožňuje jednodušší detekci chyby v případě problémů se spojením.

```
[alice]                          [bob]
type=friend                      type=friend
context=from-sip                 context=from-sip
callerid=Alice <430>             callerid=Bob <435>
secret=heslo                     secret=heslo
host=dynamic                     host=dynamic
disallow=all                     disallow=all
allow=ulaw                       allow=ulaw
qualify=yes                      qualify=yes
```

Druhá část tohoto konfiguračního souboru umožňuje vytvoření spojení mezi dvěma Asterisk ústřednami.

```
[trunk-PL]                       [trunk-CZ]
type=peer                        type=peer
```

Parametr type je v tomto případě nastaven na hodnotu peer, která reprezentuje entitu, na níž Asterisk posílá hovory.

```
host=192.168.88.105              host=192.168.88.100
```

Tato IP adresa reprezentuje IP adresu druhé ústředny Asterisk a umožňuje vytvoření spojení mezi nimi.

```
context=from-sip                 context=from-sip
username=trunk-CZ                username=trunk-PL
secret=heslojeveslo              secret=heslojeveslo
qualify=yes                      qualify=yes
```

Zbývající parametry jsou podobné s již vysvětlenou konfigurací v kapitole 4.5.1.1, navíc je pouze parametr *qualify*, který již byl také zmíněn.

4.5.3.2 ***extension.conf***

Strana CZ

```
[from-sip]
exten => 430,1,Dial(SIP/alice)
exten => _0048.,1,Set(CALLERID(num)=00420${CALLERID(num)})
exten => _0048.,2,Dial(SIP/trunk-PL/${EXTEN:4})
```

Strana PL

```
[from-sip]
exten => 435,1,Dial(SIP/bob)
exten => _00420.,1,Set(CALLERID(num)=0048${CALLERID(num)})
exten => _00420.,2,Dial(SIP/trunk-CZ/${EXTEN:5})
```

V těchto konfiguračních souborech je potřeba nastavit správně dialplan tak, aby bylo možné zavolat z jednoho SIP klienta registrovaného na CZ straně na klienta registrovaného na straně PL. První dva řádky byly již vysvětleny v kapitole 4.5.1.2. Další řádek využívá proměnnou *CALLERID(num)*, která představuje přednastavenou proměnnou obsahující číslo ID volajícího. Na začátku tohoto řádku je definováno, že toto pravidlo bude použito pro čísla začínající 00420. Dále následuje nastavení priority tohoto kroku a poté nastavení proměnné *CALLERID(num)* na tu samou hodnotu, akorát s předponou 00420, nebo 0048 v závislosti na tom, o který konfigurační soubor na které ústředně se jedná. Následující řádek představuje pravidlo se stejným začátkem, avšak prioritou dvě. V tomto případě je použita funkce Dial(), která používá jako první parametr technologii, v tomto případě SIP skrze trunk a proměnnou *${EXTEN:5}*, jež je volaným číslem, avšak v tomto případě jsou ignorována první 4 čísla. V případě konfiguračního souboru pro CZ se jedná o prvních 5 čísel.

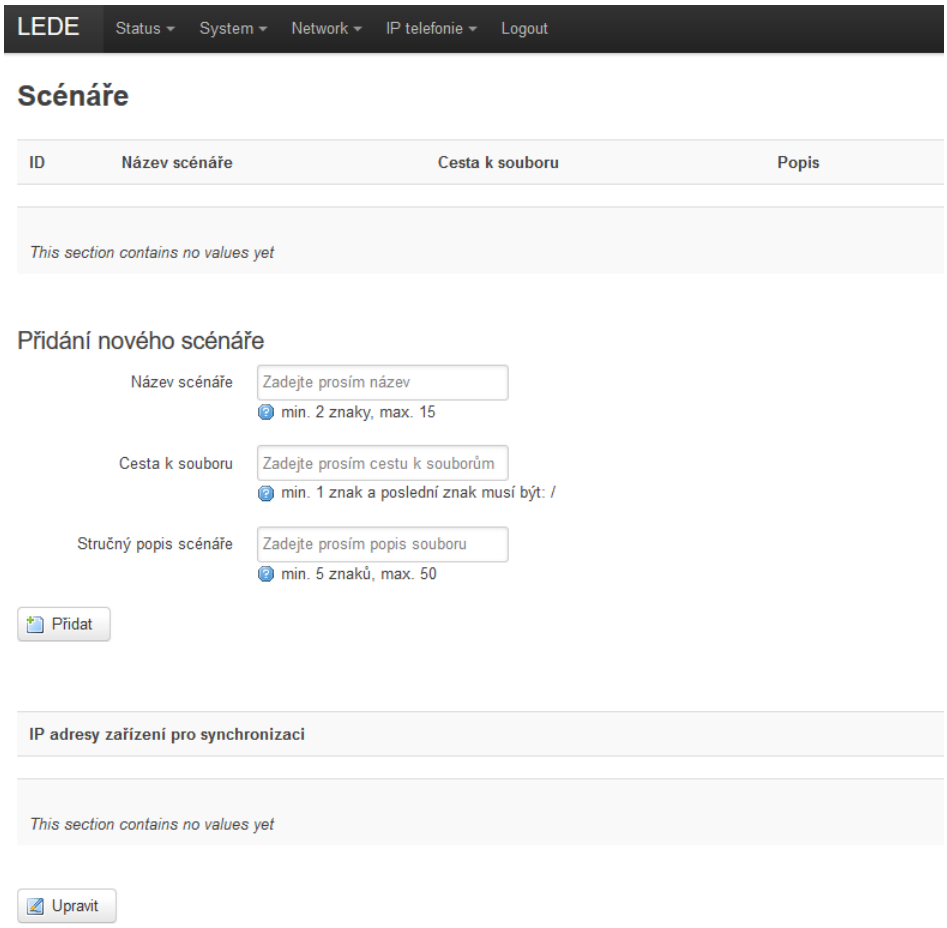
4.6 Testování celého konceptu

V rámci testování byly vyzkoušeny základní funkce a úkony, u kterých se předpokládá, že je uživatel bude vykonávat tak, aby testy byly co nejblíže skutečnému provozu.

4.6.1 Test č.1

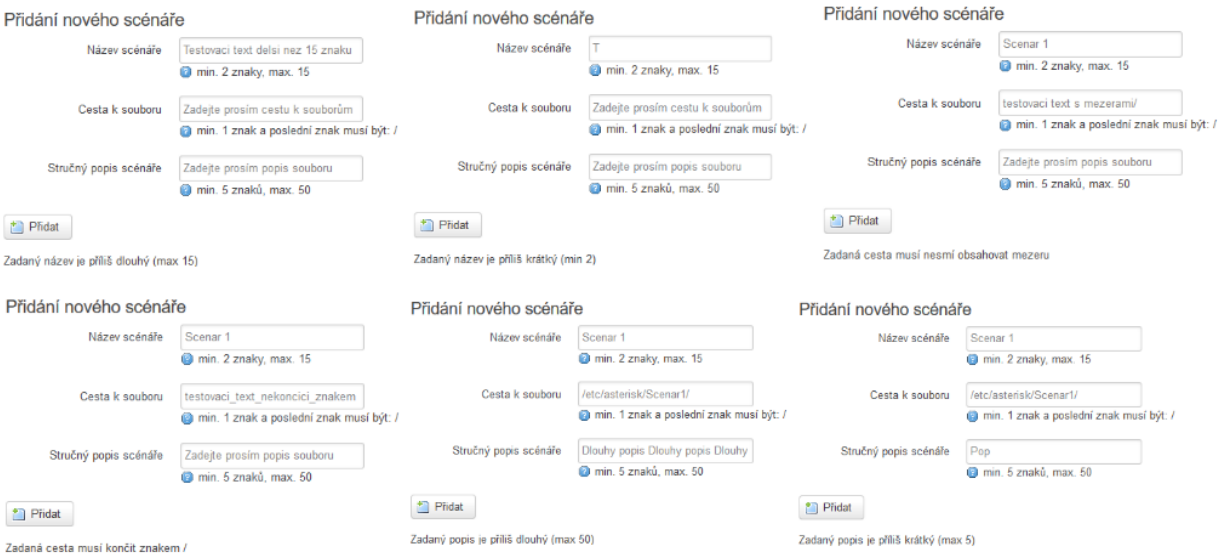
Účelem tohoto testu je otestovat, zdali správně funguje prvotní načtení obrazovek v případě chybějících pomocných souborů, a provést přidání několik IP adres a jednoho scénáře, včetně nahrání potřebných souborů a vytvoření složky s tímto scénářem. Součástí testu by také mělo být ověření funkčnosti kontrol vstupních údajů jak v případě scénáře, tak u IP adresy. Test bude proveden na operačním systému LEDE verze 17.01.4, doinstalovaným balíčkem R-sync a pouze zkopírovanými konfiguračními soubory této práce na správná místa ve stromové struktuře LuCI.

Po připojení na uživatelské rozhraní LuCI skrze IP adresu zařízení a následném přihlášení bylo rozkliknuto rozbalovací menu v horní liště. Po kliknutí na položku Scénáře se správně objevila již zmíněná hláška o chybějícím souboru *file_add.lua* s možností jeho vytvoření - Obrázek 4.6 po stisknutí tlačítka se objevila druhá hláška o chybějícím pomocném souboru *ip_add.lua* také s možností jeho vytvoření - Obrázek 4.6 . Následně se již zobrazilo standartní uživatelské rozhraní Obrázek 4.20 .



Obrázek 4.20: Uživatelské rozhraní se Scénáři

Dalším úkolem je tedy přidání nového scénáře. Nyní proběhl test kontroly vstupních dat, kdy pro každou vyžadovanou položku byl zadán chybný údaj, jak ve špatném rozsahu, tak případně se špatnou koncovkou. Ukázku některých těchto chybových hlášení je možné vidět na obrázku 4.21 .



Obrázek 4.21: Příklad chybových hlášek na vstupních údajích nového scénáře

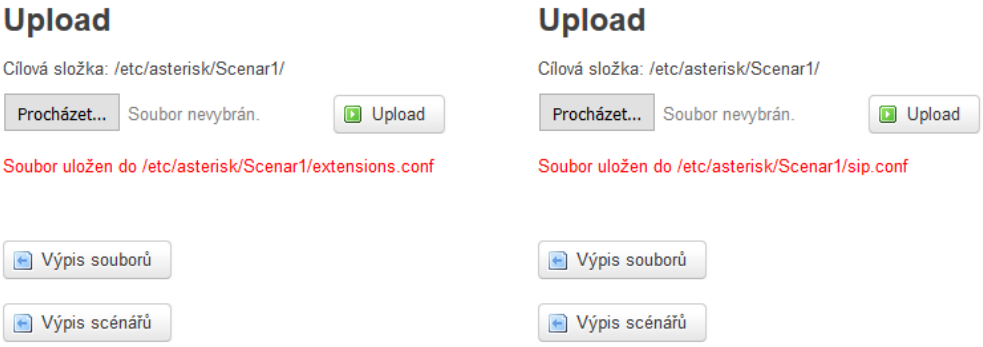
Potom, co tedy byly vloženy správné údaje, je již možné vidět údaje o tomto scénáři v tabulce na začátku stránky a tyto údaje jsou také jako výchozí nyní zobrazeny v textových polích pro zadání dalšího scénáře - Obrázek 4.22 jako příklady možných vstupních údajů.

Obrázek 4.22: Ukázka prvního vloženého scénáře

Nyní otestujeme zobrazení obsahu složky, kterou jsme zadali. Podotýkám, že tato složka zatím neexistuje, je tedy úkolem programu ji vytvořit. Také by se měl vytvořit další pomocný soubor *conf_edit.lua*. Po stisknutí tlačítka Zobrazit se objeví okno, které oznamuje, že se ve složce nenacházejí žádné konfigurační soubory - Obrázek 4.23 a vypadá to, že ani nic jiného, což je očekávaný výsledek. Je zde však možnost pro vytvoření složky či souboru. V tomto testovacím scénáři však jde o vyzkoušení funkčnosti nahrání souboru. Nyní tedy ještě ověříme skrze SSH spojení, že složka */etc/asterisk/Scenar1* byla úspěšně vytvořena, což byla.

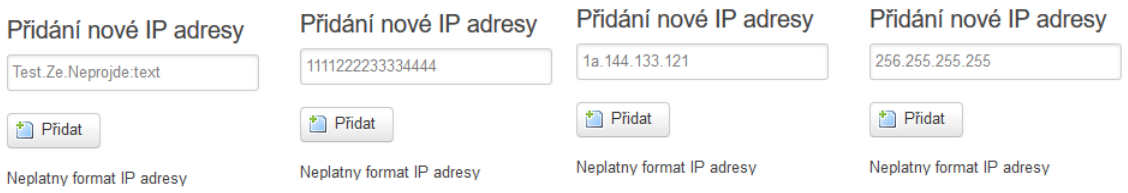
Obrázek 4.23: Ukázka zobrazení obsahu prázdné složky

Nynějším úkolem je nahrání dvou souborů, a to konfiguračních souborů pro *sip.conf* a *extension.conf*, které jsou nutné pro konfiguraci B2BUA Asterisk. Po kliknutí na tlačítko Nahrát soubor do složky je možné vidět okno s možností nahrání souborů. Zde tedy nahrajeme předpřipravené soubory *sip.conf* a *extension.conf*. V ukázce - Obrázek 4.24 je možné vidět výsledek tohoto nahrávání. Po návratu zpět na obsah složky, je možné zde tyto soubory vidět i zobrazit jejich obsah. Tato část testu tudíž proběhla také úspěšně.



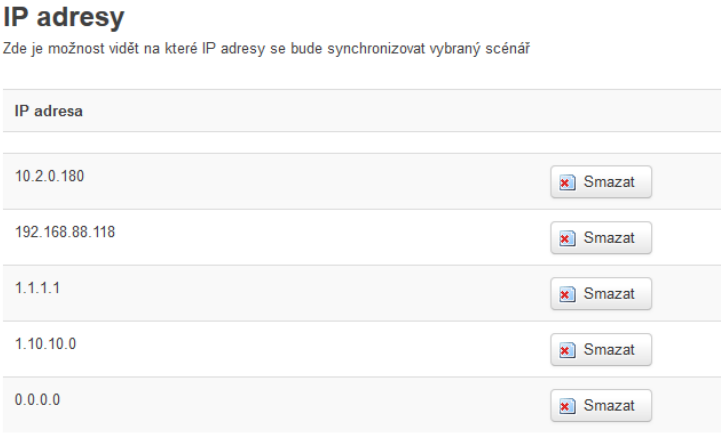
Obrázek 4.24: Ukázka oken po nahrání souboru sip.conf a extension.conf

Nyní ještě zbývá otestovat přidání IP adresy. Tentokrát tedy zvolíme v horní liště rozbalovacího menu položku IP adresy, čímž se nám zpřístupní okno s prázdnou tabulkou a možností přidat novou IP adresu. Nyní tedy otestujeme kontrolu vstupních hodnot IP adresy, přičemž očekáváme, že kontrolou projdou pouze standartní IP adresy verze 4. Na obrázku 4.25 je možné vidět výsledek několika špatně zadaných formátů IP adresy. U všech byla zobrazena chybová hláška o neplatnosti tohoto formátu.



Obrázek 4.25: Ukázka několika špatně zadaných vstupních dat pro IP adresu

Následuje ukázka několika správně zadaných IP adres tak, jak se zobrazily v tabulce pro IP adresy. Následně bylo vyzkoušeno i smazání IP adresy, což se obešlo bez problému.



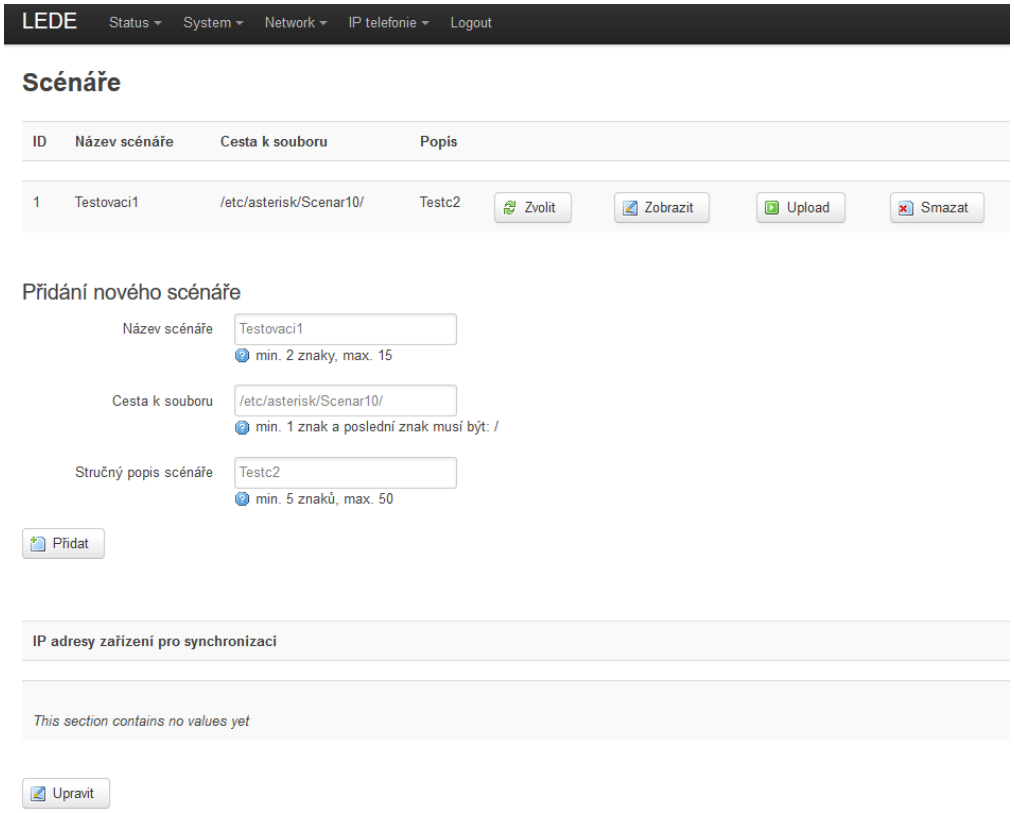
Obrázek 4.26: Ukázka některých správně zadaných IP adres

Test č.1 tedy proběhl dle očekávání a je tedy možné ho hodnotit za úspěšně provedený.

4.6.2 **Test č.2**

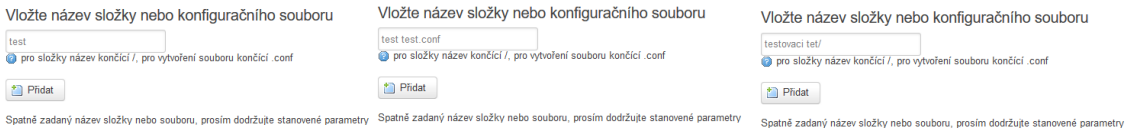
Tento test ověřuje funkčnost možnosti vytvoření složky a konfiguračního souboru, přičemž by mělo dojít také k uložení některých dat do tohoto souboru. Následuje ověření, zdali dojde skrze hlavní okno se scénáři ke smazání testovacího scénáře a všech souborů a složek souvisejících s tímto scénářem.

Po připojení na uživatelské rozhraní v sekci Scénáře byl vytvořen scénář s názvem Testovaci1 - Obrázek 4.27 , jenž je v tomto testu použit.



Obrázek 4.27: Ukázka vloženého testovacího scénáře

Po stisknutí tlačítka Zobrazit u tohoto scénáře dojde k přesměrování na okno se zobrazením obsahu složky 4.4.3, ta je nyní prázdná. Avšak nyní ověříme, zdali je možné vytvořit pouze soubory či složky dle požadovaných parametrů. Pro vytvoření složky je potřeba název ukončit znakem / a pro vytvoření konfiguračního souboru koncovkou *.conf*. Nejsou povoleny mezery v názvu a lze vytvořit i strukturu několika složek, například zadáním názvu */test1/test2/test3* dojde k vytvoření složky test3 nacházející se ve složce test2, která je obsažena ve složce test1 a nachází se ve zvolené složce. Na obrázku 4.28 je možné vidět několik špatně zadaných vstupních dat, přičemž pokaždé došlo k zobrazení hlášky o tom, že zadané údaje jsou chybné.

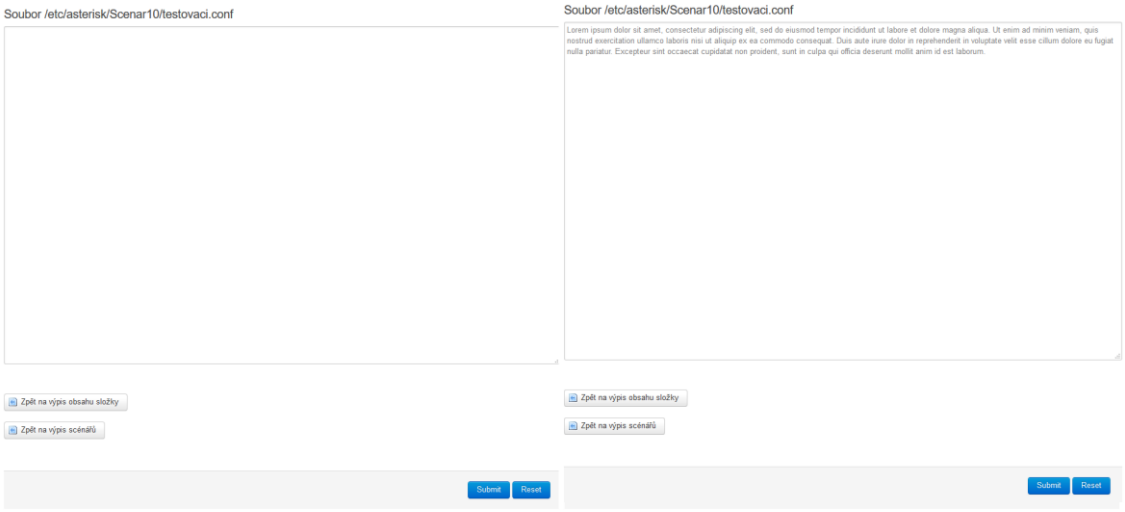


Obrázek 4.28: Ukázka některých chybně zadaných vstupních dat CBI modelu edit_scenare.lua

Následně byl v rámci testování vytvořen jeden soubor a jedna složka, které již splňovaly požadované parametry - Obrázek 4.29 .

Obrázek 4.29: Ukázka vytvořeného souboru a složky v CBI modelu edit_scenar.lua

Nyní je potřeba otestovat, zdali je možné vytvořený soubor editovat a data do něj uložit. Kliknutím na tlačítko Editovat došlo k přesměrování na okno s možností editace vytvořeného souboru, jenž je zatím prázdný, jak je možné vidět na levé polovině - Obrázek 4.30 . Na pravé polovině je možné vidět soubor se zadaným textem, který od něj byl úspěšně uložen po stisknutí tlačítka Submit.



Obrázek 4.30: Ukázka testovacího souboru otevřeného v CBI modelu save_conf.lua

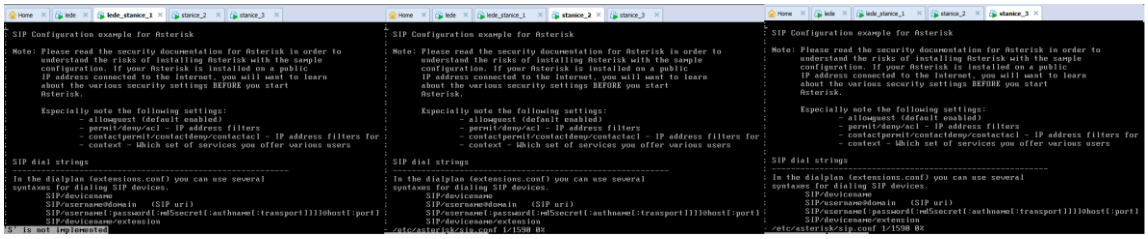
Během testu tedy byla vytvořena složka a soubor, jenž byl i naplněn daty. Ověření, že opravdu došlo k vytvoření výše zmíněného souboru a složky skrze SSH spojení. Nyní již zbývá pouze otestovat, zdali budou tento soubor i složka, včetně nadřazené složky, smazány po stisknutí

tlačítka Smazat na hlavním okně se scénáři 4.4.1 a ověřit tuto skutečnost přes SSH spojení. Po návratu na hlavní okno a po stisknutí příslušného tlačítka opravdu došlo ke smazání všech požadovaných dat. Test tudíž proběhl úspěšně.

4.6.3 **Test č. 3**

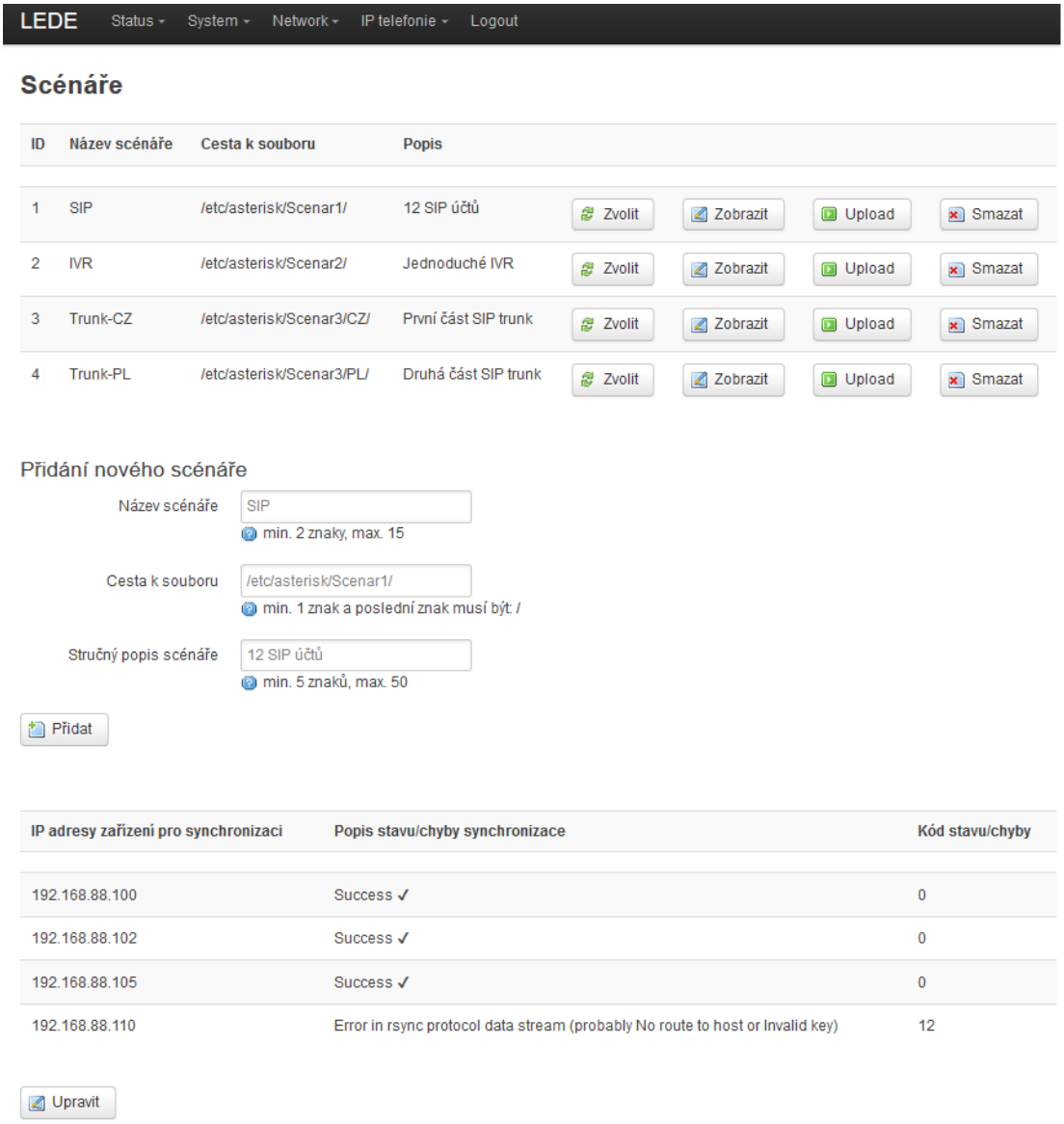
Test má za cíl ověřit funkčnost synchronizace scénáře s několika vzdálenými stanicemi, které jsou dostupné v rámci sítě, a správné zobrazování výsledku této synchronizace. Také ověřit, zdali je synchronizovaný scénář funkční. Součástí tohoto testu je tedy otestování funkčnosti všech připravených scénářů 4.5.

Nejprve bylo potřeba předpřipravit a spustit virtuální stanice. Následně zjistit jejich IP adresy a tyto vložit do tabulky s IP adresami 4.4.2, přičemž do tabulky byla vložena i IP adresa stroje, který není spuštěn, a tudíž není v rámci sítě dostupný, aby bylo ověřeno, že bude vygenerována chybová hláška. Pro tento test byly použity tři stanice se stejným nastavením provedeným dle kapitol 4.1 až 4.3. Také scénáře již byly předpřipraveny a obsahují konfigurační soubory.



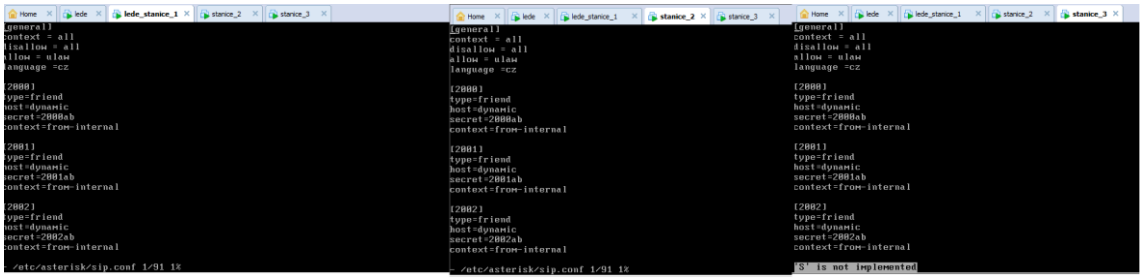
Obrázek 4.31: Konfigurační soubory sip.conf na jednotlivých stanicích před synchronizací

Po stisknutí tlačítka Zvolit u scénáře s názvem SIP došlo k synchronizaci vzdálených stanic. Výsledek je možné vidět na obrázku 4.32 . Z něho vyplývá, že synchronizace proběhla úspěšně u všech IP adres až na jednu, která byla právě zmiňovanou testovací a nebyla dostupná.



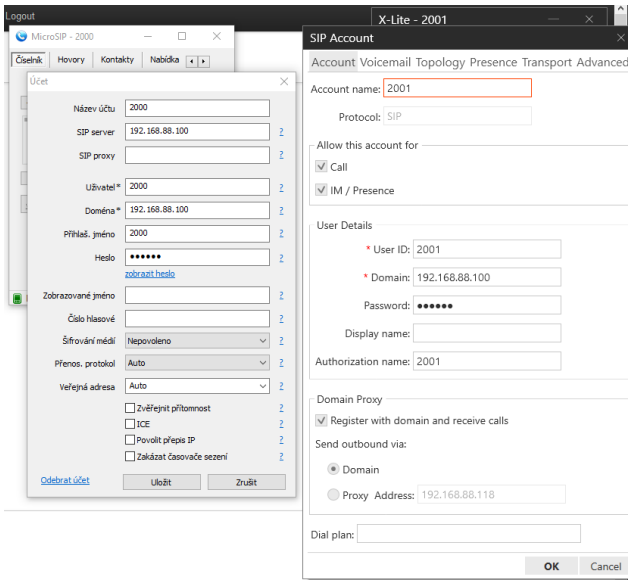
Obrázek 4.32: Ukázka výsledku synchronizace v rámci testu č.3

Tento výsledek byl také ověřen skrze zobrazení obsahu konfiguračního souboru *sip.conf* nacházejícího se na jednotlivých stanicích - Obrázek 4.33 , které bylo možné porovnat s původním obsahem před synchronizací - Obrázek 4.31 .



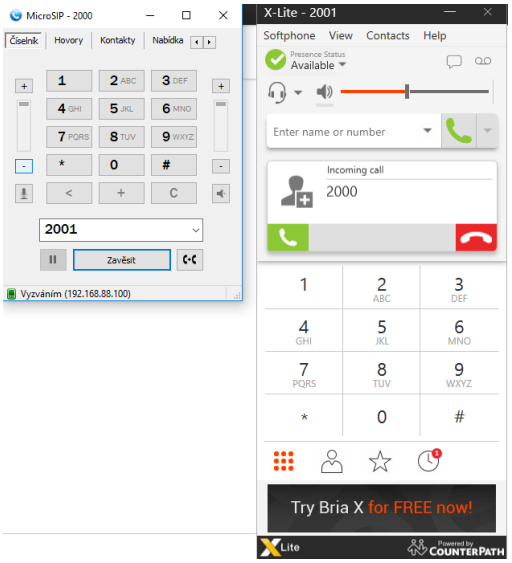
Obrázek 4.33: Konfigurační soubory sip.conf na jednotlivých stanicích po synchronizaci

Dalším ověřením je otestování funkčnosti tohoto scénáře na všech stanicích. K tomuto účelu byly jako SIP klienti použity programy MicroSIP [17] a X-Lite [17]. Ukázku jejich konfigurace je možné vidět na obrázku 4.34 . Každý klient byl připojen na jiný účet, který je nadefinován v rámci konfiguračního souboru *sip.conf* 4.5.1.1 u každé stanice. Ověření funkčnosti proběhlo v rámci všech testovaných stanic, konfigurace SIP klientů se lišila pouze ve změně IP adresy.



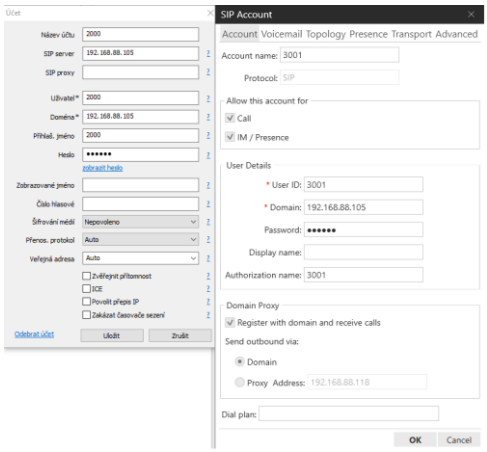
Obrázek 4.34: Ukázka konfigurace SIP klientů

K ověření došlo tím, že proběhla úspěšná registrace klienta a následně proběhl testovací hovor mezi těmito dvěma klienty - Obrázek 4.35 . Všechny tyto testovací hovory proběhly v pořádku, tudíž lze tento test považovat za úspěšný.



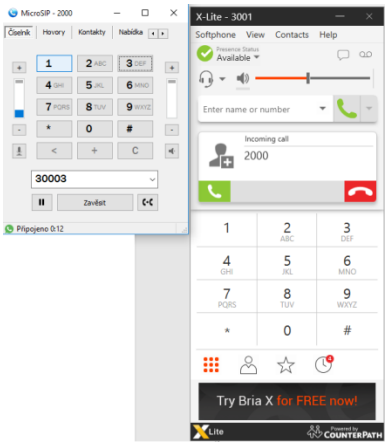
Obrázek 4.35: Ukázka testování spojení mezi dvěma SIP klienty

Nyní došlo na testování scénáře číslo dva 4.5.2, jenž představuje systém IVR 4.5.2. K účelu testování tohoto scénáře zůstaly stejné stanice, pouze došlo k synchronizaci souborů scénáře IVR a přenastavení klientů. Přitom první klient posloužil jako volající a druhý jako operátor pod číslem 3001 - Obrázek 4.36 .



Obrázek 4.36: Ukázka konfigurace klientů pro scénář IVR

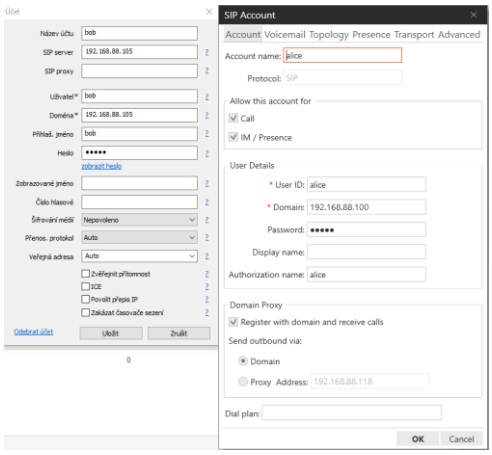
Následně došlo k několika testovacím hovorům tak, aby bylo ověřeno, že všechny položky menu fungují správně. Na obrázku 4.37 je možné vidět ukázku, kdy bylo voláno na číslo 3000, které reprezentuje hlavní menu navrženého IVR. V tomto menu byla vybrána hodnota 3, jež znamená požadavek na spojení s operátorem. Po stisknutí tohoto tlačítka tedy bylo voláno na číslo 3001 registrované na druhém klientovi. Jak je možné vidět, i tento test proběhl úspěšně a došlo ke spojení. Tudíž byla ověřena funkčnost tohoto scénáře.



Obrázek 4.37: Ukázka testovaného hovoru v rámci scénáře IVR

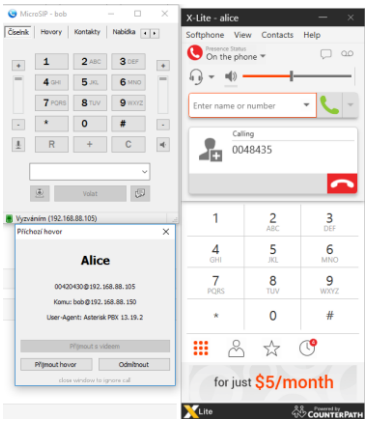
Posledním scénářem k testu je scénář SIP Trunk 4.5.3, jenž představuje vytvoření spojení mezi dvěma ústřednami, které se mohou nacházet na dvou různých místech tak, aby bylo umožněno volání mezi těmito ústřednami. Tento scénář bylo potřeba synchronizovat minimálně dvakrát, jelikož se jedná o vytvoření spojení mezi dvěma ústřednami; je potřeba nejprve synchronizovat první část scénáře s vybranými stanicemi, které budou sloužit jako ústředna CZ,

a poté až druhou část s dalšími stanicemi, které budou představovat ústřednu PL. Také je potřeba změnit konfiguraci klientů - Obrázek 4.38 .



Obrázek 4.38: Ukázka konfigurace klientů pro scénář SIP trunk

Po synchronizaci a přenastavení klientů tedy došlo na samotný test, přičemž nejprve proběhl test pro první pár testovaných ústředen a posléze pro druhý. Pro první proběhlo vše dle očekávání - Obrázek 4.39 . Při následné změně konfigurace klientů však již nedošlo ke spojení. Chyba byla v konfiguračním souboru *sip.conf* 4.5.3.1, v němž je v případě tohoto scénáře potřeba nastavit správné IP adresy vzdálených ústředen. Po změně konfigurace souboru *sip.conf* na správné IP adresy již test proběhl v pořádku.



Obrázek 4.39: Ukázka testovaného spojení pro scénář SIP trunk

Závěr

Jak je možné vidět i z výsledků jednotlivých testů, vývoj uživatelského rozhraní dle zadání byl úspěšný a věřím tomu, že jeho nasazení povede k zefektivnění výuky související s IP telefonní na VŠB-TU Ostrava a také vyučujícím usnadní jejich nelehkou práci. Vytvořené řešení bylo otestováno na posledních dvou verzích OpenWRT/LEDE v kombinaci se SIP B2BUA Asterisk verze 13, proto by měla být zaručena dostatečná kompatibilita i dostatečná aktuálnost tohoto řešení. Vývoj samotného systému zabral spoustu času, ale i tak je zde mnoho věcí, které se dají vylepšit, udělat jinak či přidat nové funkce. Mohu uvést například možnost výběru jen některých IP adres z tabulky pro synchronizaci nebo propojení tohoto projektu s jiným, který v tomto roce začal zpracovát jiný student a jeho práce se věnuje vytvoření GUI pro práci s Asterisk. Propojení těchto dvou systémů by umožnilo jednodušší editaci konfiguračních souborů pro Asterisk a rovněž urychlilo vytváření nových scénářů. Mé řešení by se také dalo poněkud pozměnit a místo pomocných souborů použít konfigurační soubory, vytvořené speciálně pro LuCI, a ty propojit s použitím třídy *Map* na místo *SimpleForm*, která je používána v tomto projektu. Další možností rozšíření by bylo přidání podpory pro IP verze 6. Bylo určitě velmi zajímavé i náročné naučit se pracovat v programovacím jazyce Lua v kombinaci s některými zvláštnostmi a specifiky danými jeho použitím v rámci LuCI. Věřím, že tato diplomová práce bude nápomocna i dalším studentům, kteří se budou zabývat podobnou problematikou, a opravdu mě těší, že tento projekt bude mít reálné využití v rámci výuky.

Použitá literatura

[1] Obecná veřejná licence GNU v.3 (GNU GPL v.3): GNU General Public License. PETÁK, Tibor. *GNU General Public License: Překlad do češtiny*[online]. 2007 [cit. 2017-12-04]. Dostupné z: http://www.gnugpl.cz/v3/

[2] BusyBox. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-12-04]. Dostupné z: https://cs.wikipedia.org/wiki/BusyBox

[3] Licence MIT. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-12-04]. Dostupné z: https://cs.wikipedia.org/wiki/Licence_MIT

[4] *Kamailio: The Kamailio SIP Server Project* [online]. [cit. 2017-12-04]. Dostupné z: https://www.kamailio.org/w/

[5] BRYANT, Stephen. Asterisk: the definitive guide. 4. vyd. Sebastopol: O´Reilly, 2013. ISBN 978-1449332426.

[6] IERUSALIMSCHY, Roberto. Programming in Lua, Fourth Edition. Fourth Edition. 2016.

[7] OpenWrt on VMware HowTo [online]. In: . [cit. 2018-03-03]. Dostupné z: https://wiki.openwrt.org/doc/howto/vmware?s%5B%5D=vmware

[8] OpenWRT: Command-line interpreter [online]. In: . [cit. 2018-03-03]. Dostupné z: https://wiki.openwrt.org/doc/howto/user.beginner.cli

[9] VMWare Workstation User's Manual.[online][cit. 2018-03-04]. Dostupné z: http://pubs.vmware.com/ws7_ace26/wwhelp/wwhimpl/js/html/wwhelp.htm?context=ws_user&file=ws_net_configurations_bridged.html

[10] HOLT, Alan. Embedded operating systems: a practical approach. London: Springer, 2014. Undergraduate topics in computer science. ISBN 978-1447166023.

[11] Qemu-img for Windows. Cloudbase solution [online]. [cit. 2018-03-05]. Dostupné z: https://cloudbase.it/qemu-img-windows/

[12] Index of (root) / releases / 17.01.4 / targets /. Lede-project [online]. [cit. 2018-03-05]. Dostupné z: https://downloads.lede-project.org/releases/17.01.4/targets/

[13] 佐须之男. Wifidog OpenWrt luci页面. WiFiDog - A Captive Portal Suit - 无线热点认证解决方案 [online]. 2015 [cit. 2018-03-05]. Dostupné z: http://www.wifidog.pro/2015/01/04/wifidog-luci.html

[14] HowTo: Write Modules: Luci. GitHub [online]. [cit. 2018-03-06]. Dostupné z: http://luci.subsignal.org/trac/wiki/Documentation/ModulesHowTo

[15] Module nixio.fs [online]. [cit. 2018-03-08]. Dostupné z: https://neopallium.github.io/nixio/modules/nixio.fs.html

[16] Rsync exit codes [online]. [cit. 2018-03-09]. Dostupné z: https://lxadm.com/Rsync_exit_codes

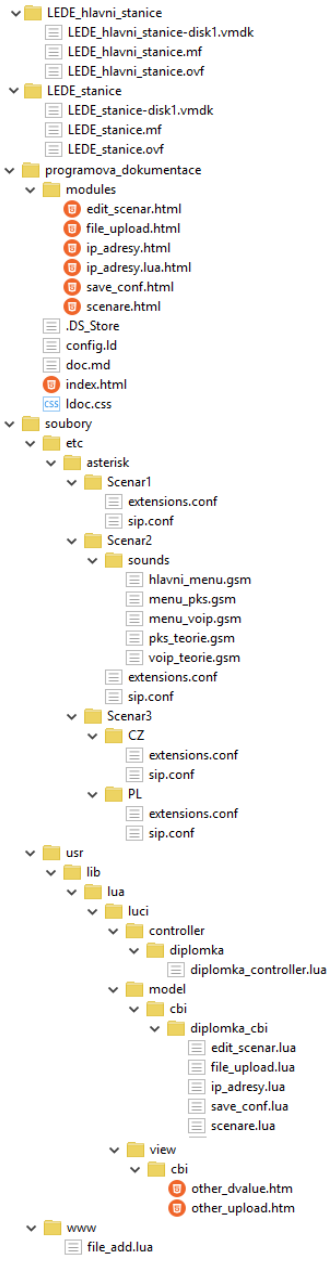
[17] Microsip [online]. MDev Group [cit. 2018-03-10]. Dostupné z: https://www.microsip.org/

[18] X-Lite - Welcoming You to the World of Softphones [online]. CounterPath, 2018 [cit. 2018-03-10]. Dostupné z: http://www.counterpath.com/x-lite/

Seznam příloh

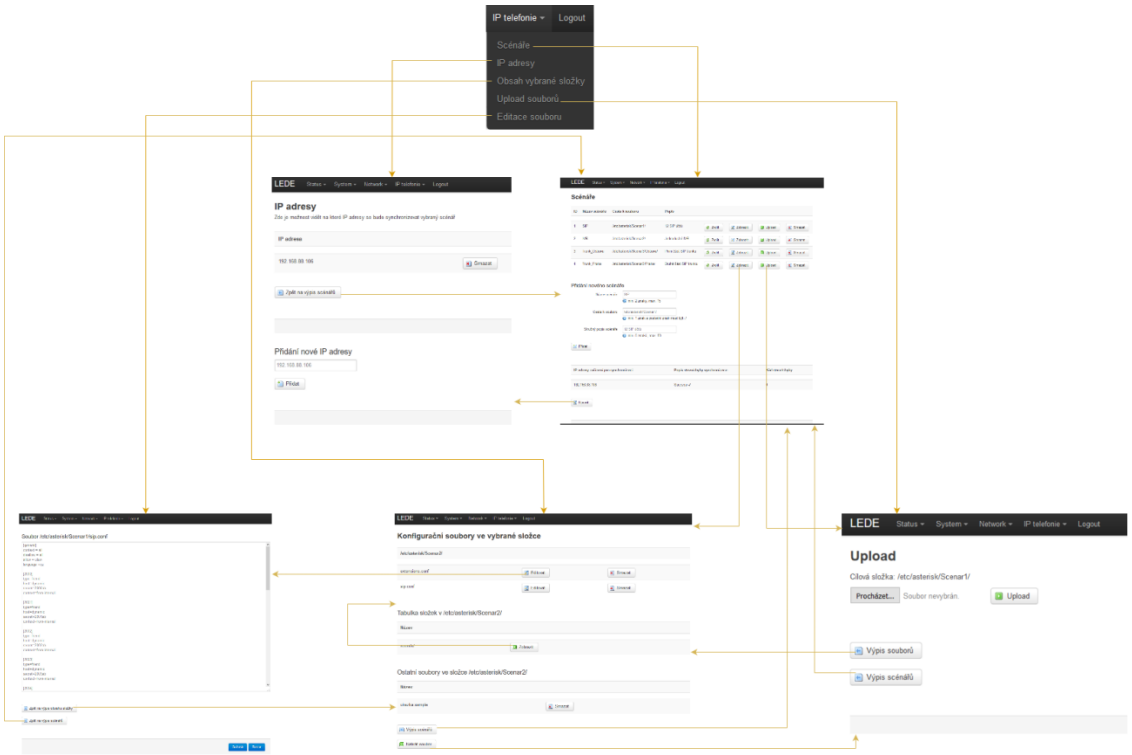
Součástí BP/DP je CD/DVD.

Adresářová struktura přiloženého CD/DVD:



Příloha A: *Uživatelská dokumentace*

Účelem této části je seznámit se se způsoby ovládaní, jeho funkcemi a možnostmi tak, aby uživatel věděl, jaký výsledek má od té či jiné funkce očekávat a mohl tedy uživatelské rozhraní pohodlně používat. I z důvodu lepší přehlednosti je součástí této uživatelské dokumentace navigační mapa - obrázek 1.1, která zobrazuje odkud a kam se lze dostat skrze horní menu či tlačítka.



Obrázek 1.1: *Navigační mapa*

Jako jeden z možných ovládacích prvků je kontextové menu, které je v horní liště stránky. Po najetí ukazatelem myši na toto menu se zobrazí několik položek - obrázek 1.2 , které reprezentují jednotlivá uživatelská rozhraní a je možné na ně kliknutím přejít. Položka (1) přesměruje uživatele na hlavní okno se scénáři, položka (2) na okno pro úpravu IP adres, třetí položka v pořadí (3) na uživatelské rozhraní, ve kterém se zobrazuje obsah složky, další pak na možnost nahrání souboru (4) a poslední na uživatelské okno, které je určeno pro editaci konfiguračních souborů (5).

Obrázek 1.2: *Kontextové menu*

A.1 Popis hlavní obrazovky se scénáři

Po instalaci, resp. zkopírování, souborů avšak mimo, nejsou ještě vytvořeny pomocné soubory, tudíž se obrazovky jednotlivých oken poněkud liší; stejná situace nastane, pokud některé z pomocných souborů nejsou z nějakého důvodu dostupné. V takovém případě mohou u tohoto okna nastat dvě situace - obrázek 1.3 . První je zobrazení chybové hlášky o chybějícím pomocném souboru file_add.lua (1) s tlačítkem (2), po jehož stisknutí se požadovaný soubor vytvoří. Druhou možností je velice podobné okno, jen s jinou chybovou hláškou (3) o chybějícím pomocném souboru ip_add.lua a tlačítkem (4), jehož stisknutí nyní vytvoří soubor ip_add.lua.

Obrázek 1.3: *Uživatelské rozhraní Scénáře v případě chybějících pomocných souborů*

Pokud jsou soubory již vytvořeny obrazovka - obrázek 1.5 má sloužit jako hlavní ovládací panel, kde má uživatel přístup ke vše hlavním funkcím. Horní lišta (1) je rozbalovací menu, ze kterého je možné se dostat na všechna dostupná okna. Tabulka (2) se scénáři je zobrazení všech dostupných scénářů, které byly uživatelem zadány. U každého scénáře je několik tlačítek. První z nich je tlačítko Zvolit (3), které po stisknutí provede synchronizaci se zařízeními, jejichž IP adresy se nalézají v tabulce (9). Zde je také následně zobrazeno, zdali synchronizace proběhla v pořádku, případně chyba, ke které došlo. Bohužel může nastat situace, kdy se zobrazí chyba - obrázek 1.4 . Tato situace nastane v případě, že jedna ze zadaných IP adres je mimo rozsah sítě, ve které je vzdálené zařízení. V tomto případě je potřeba se tlačítkem Zpět v prohlížeči vrátit na předchozí stránku a zkontrolovat, zda jsou všechny IP adresy v pořádku.

Obrázek 1.4: *Chyba Bad Gateway*

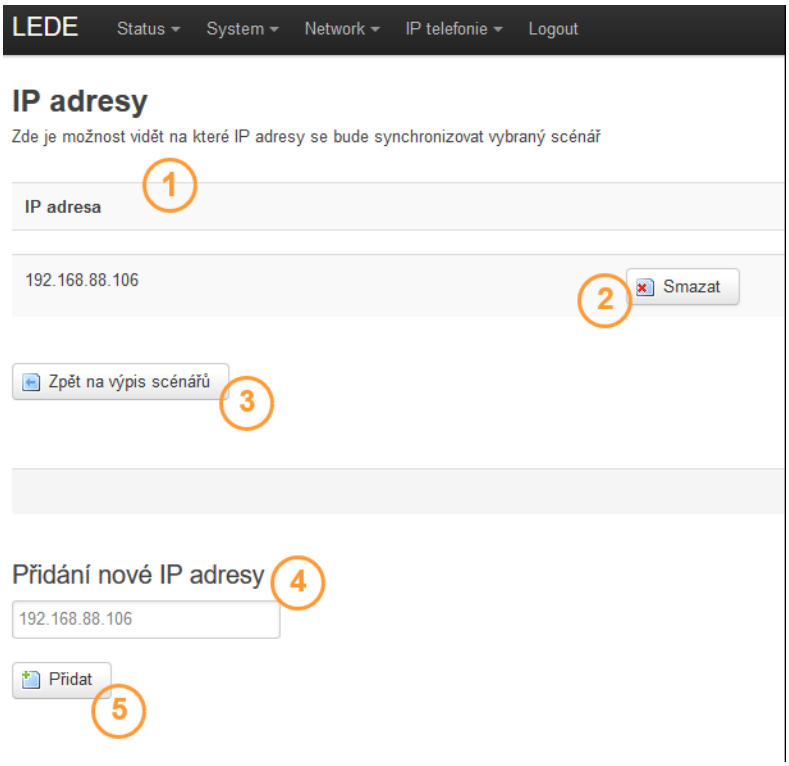
Druhým tlačítkem je Zobrazit (4). Toto tlačítko slouží pro přesměrování na okno, kde je možné zobrazit obsah složky vybraného scénáře. Pokud tato složka neexistuje, toto tlačítko požadovanou složku vytvoří. Dalším tlačítkem v pořadí je tlačítko Upload(5). Toto tlačítko slouží k přesměrování na okno s možností nahrání souboru do složky vybraného scénáře. Pokud požadovaná složka neexistuje, bude vytvořena. Posledním tlačítkem je tlačítko Smazat (6), které po stisknutí vymaže tento scénář ze seznamu včetně složky, která je u něj uvedena. Pokud složka

nebude nalezena nebo dojde k jiné chybě, zobrazí se chybová hláška a scénář nebude vymazán. Pokud jste tedy vytvořili scénář například se špatným názvem a chcete ho smazat a nemáte ještě nijak vytvořenou příslušnou složku, je potřeba kliknout na tlačítko Zobrazit (4), čím bude příslušná složka vytvořena, a následně se vrátit zpátky do okna se scénáři a scénář smazat. V další části tohoto ovládacího okna se nacházejí textová pole (7), do nichž je možné zadat údaje o novém scénáři. Je potřeba zadat všechny tři položky podle požadavků, které se nacházejí pod každým tímto textovým polem. Přitom však nezapomenout, že cesta k souboru nesmí obsahovat mezery. V opačném případě se po stisknutí tlačítka Přidat (8) zobrazí pod tímto tlačítkem text s chybovou hláškou a scénář nebude uložen. V neposlední řadě je zde již zmíněná tabulka s IP adresami (9), se kterými proběhne synchronizace, a pod touto tabulkou tlačítko Upravit (10), jehož účelem je přesměrování na okno s možnostmi práce s jednotlivými IP adresami.

Obrázek 1.5: *Popis uživatelského rozhraní Scénáře*

A.2 Okno pro práci s IP adresami

Toto uživatelské rozhraní slouží pro správu IP adres. V tabulce (1) je možné vidět aktuálně uložené IP adresy. U každé této IP adresy se na řádku nachází tlačítko Smazat, které inicializuje funkci pro odstranění vybrané IP adresy z tabulky. Pod tabulkou se nachází tlačítko (3), po jehož stisknutí bude načteno hlavní okno s jednotlivými scénáři. Dalším prvkem na tomto rozhraní je textové pole (4), do kterého je možné zapsat IP adresu. IP adresa musí být ve standardním formátu IP adresy verze 4. Tato skutečnost je i následně ověřena po stisknutí tlačítka Přidat (5), čímž se IP adresa vloží to tabulky (1). Pokud však je IP adresa v textovém poli zadána špatně, zobrazí se hláška o této skutečnosti a IP adresa nebude uložena.



Obrázek 1.6: *Ukázka uživatelského rozhraní IP adresy*

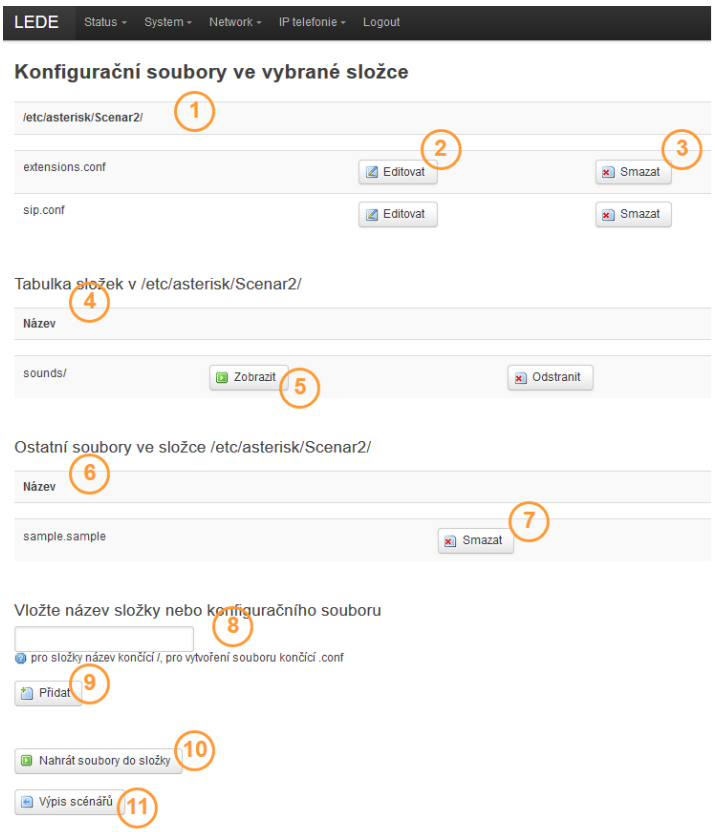
V tomto uživatelském rozhraní může také nastat situace, kdy nebude nalezen pomocný soubor ip_add.lua, jehož přítomnost toto uživatelské rozhraní vyžaduje. Pokud tato skutečnost nastane, uživatelské rozhraní bude vypadat stejně jako na pravé straně na obrázku 1.3 .

A.3 Okno pro zobrazení obsahu složky

Toto okno je rozděleno do několika sekcí a slouží pro zobrazení souborů či složek, které obsahuje vybraná složka, případně pro práci s nimi. Je zde také u každé sekce vidět, o jaký typ dat se jedná a z jaké složky jsou. Toto okno může mít vzhled jako na obrázku 1.7 v případě, že není dostupný pomocný soubor scenar_edit.lua, jen je pro fungování tohoto okna nezbytný. Tento pomocný soubor je vytvořen po kliknutí na tlačítko Zobrazit nebo Upload nacházejícím se na hlavním okně se scénáři Kapitola A.1.

Obrázek 1.7: *Uživatelské rozhraní Konfigurační soubory v případě chybějícího pomocného souboru scenar_edit.lua*

V první sekci, viz obrázek 1.8, je možné nalézt tabulku (1), ve které jsou zobrazeny veškeré konfigurační soubory, tedy ty, jejichž přípona je conf. Každý z těchto souborů je možné editovat po stisknutí tlačítka Editovat (2), které přesměruje uživatele do uživatelského rozhraní určeného k editaci. Druhé tlačítko Smazat (3), jak už název napovídá, je určeno pro smazání vybraného souboru.



Obrázek 1.8: *Uživatelské rozhraní Konfigurační soubory ve vybrané složce*

Druhá sekce slouží pro zobrazení jednotlivých složek, které jsou ve vybrané složce obsaženy. U každé této složky je tlačítko Zobrazit (5), které uživateli po stisknutí zobrazí obsah této složky. Tato sekce nemusí být vždy zobrazena, závisí to od toho, zdali se v dané složce nachází nějaká jiná složka; v případě, že nikoliv, tato sekce zůstane skryta.

Další sekce obsahuje v tabulce (6) soubory, které mají jinou koncovku než-li .conf. U těchto souborů je možné nalézt tlačítko Smazat (7), které tento soubor smaže. Stejně jako

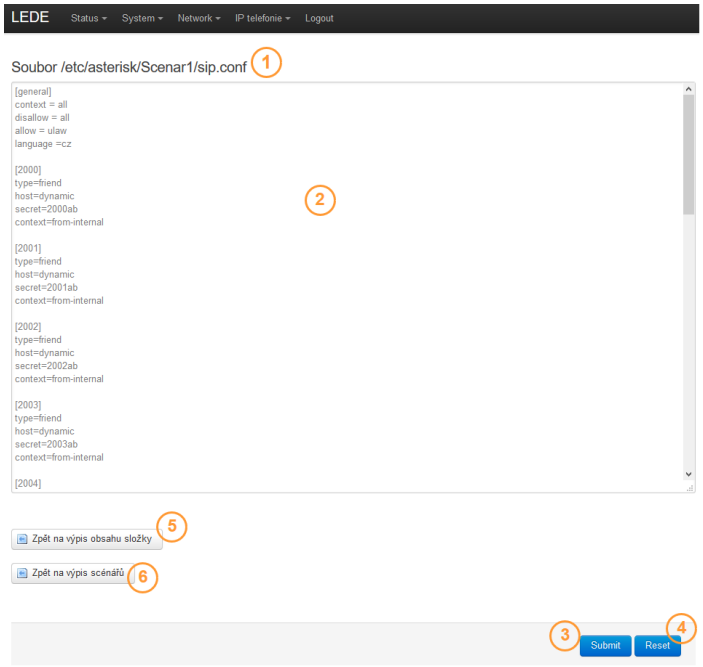
předchozí sekce i tato sekce se zobrazuje pouze, pokud se ve vybrané složce nacházejí nějaké soubory, které nemají příponu .conf.

Následuje sekce (8), ve které je možnost vytvořit soubor nebo složky, které budou vytvořeny v příslušné složce. Pokud chce uživatel vytvořit soubor, je potřeba, aby jako příponu zadal .conf; je totiž možné vytvářet pouze konfigurační soubory; název nesmí obsahovat mezeru a nesmí se shodovat s názvem, který se již nalézá v této složce. Podobná pravidla platí pro vytvoření složky, zde však název musí končit znakem /. Po stisknutí tlačítka (9) je provedena kontrola vstupních dat. V případě, že uživatel zadá něco chybně, je o tom vyrozuměn. Pokud je vše v pořádku, dojde k vytvoření složky nebo souboru.

V poslední sekci se nacházejí dvě navigační tlačítka. První tlačítko (10) přesměruje uživatele na hlavní okno se scénáři a druhé (11) na uživatelské okno pro nahrání souboru.

A.4 Okno pro úpravu konfiguračních souborů

Uživatelské rozhraní sloužící pro editaci souborů s příponou .conf (obrázek 1.9). V rámci názvu tohoto okna je popsána úplná cesta k zobrazenému souboru (1). Následuje textové pole o délce 30 řádků (2), ve kterém je zobrazen obsah tohoto souboru. Tato data je možné editovat a tlačítkem Submit (3) úpravy uložit. Pokud je potřeba změny neukládat a vrátit zpět, stačí stisknout tlačítko Reset (4). Dále se již na tomto rozhraní nacházejí pouze dvě navigační tlačítka; tlačítko (5), které přesměruje uživatele na uživatelské rozhraní zobrazující obsah složky, ve které se tento soubor nachází, a druhé (6) pro návrat na do hlavního okna, kde jsou zobrazeny všechny scénáře.



Obrázek 1.9: *Uživatelské rozhraní pro úpravu vybraného souboru*

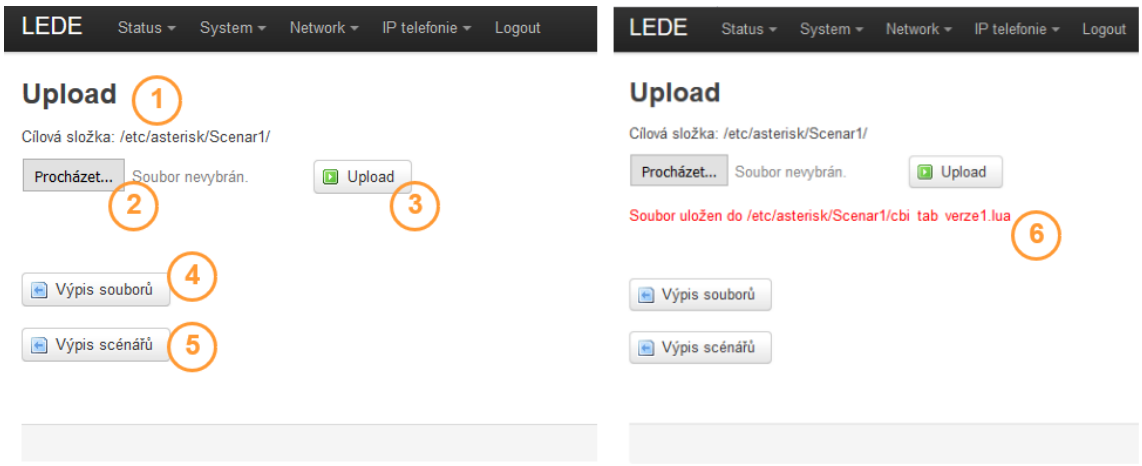
Toto uživatelské rozhraní nemusí být dostupné, pokud chybí jeden z pomocných souborů scenar_edit.lua a conf_edit.lua. Poté bude rozhraní vypadat jako jedno z oken na obrázku 1.10. Je

tedy potřeba přejít na jiné okno v závislosti na tom, který soubor chybí; veškeré informace jsou rovněž napsány v samotném okně rozhraní.

Obrázek 1.10: *Uživatelské rozhraní při chybějících pomocných souborech*

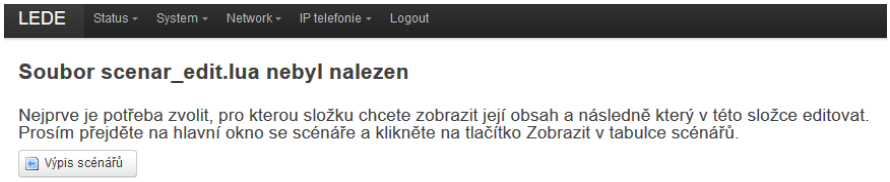
A.5 Okno pro nahrání souboru

Toto uživatelské rozhraní (obrázek 1.11) umožňuje uživateli nahrát jakýkoliv soubor do vybrané složky, jejíž cesta je popsána pod nadpisem Upload (1). Pod tímto se nachází tlačítko Procházet (2), po jehož stisknutí se zobrazí standardní systémové okno pro procházení složek a souborů, určené pro nalezení požadovaného souboru. Po jeho vybrání se jeho název zobrazí místo textu Soubor nevybrán. Pod tímto názvem bude také nahrán do uvedené složky. Samotné nahrání souboru se provede tlačítkem Upload, kdy po nahrání souboru se zobrazí hláška o tom, zda a kam byl požadovaný soubor nahrán (6). V případě nějaké chyby během nahrávání se zde zobrazí hláška s touto chybou.



Obrázek 1.11: *Uživatelské rozhraní Upload*

Další tlačítka jsou určena pro navigaci mezi jednotlivými okny. První tlačítko (4) provede přesměrování na okno s výpisem obsahu složky a druhé (5) na hlavní okno se všemi scénáři. Toto okno může vypadat jako na obrázku v případě chybějícího pomocného souboru scenar_edit.lua; v takovém případě je potřeba se držet napsaných instrukcí a pomocný soubor vytvořit.



Obrázek 1.12: *Uživatelské rozhraní Upload v případě chybějícího pomocného souboru*

Příloha B: *Programová dokumentace*

Programová dokumentace byla vygenerována pomocí programu Ldoc, který je možné nalézt na odkazu https://stevedonovan.github.io/ldoc/. Tato programová dokumentace se nachází ve složce programova_dokumentace ve stromovém adresáři na přiloženém CD . Stejně tak je zde možné nalézt všechny soubory nutné pro instalaci a zprovoznění vytvořeného systému. Na CD se také nalézají vyexportované virtuální stroje a to ve složkách LEDE_hlavni_stanice a LEDE_stanice.